Release ZERO.0.1 of package RefereeToolbox

Frédéric Dambreville
http://email.fredericdambreville.com

September 26, 2013

1 Description

RefereeToolbox is a java package implementing combination operators for fusing evidences. It is downloadable from:

http://refereefunction.fredericdambreville.com/releases

RefereeToolbox is based on an interpretation of the fusion rules by means of Referee Functions (refer to 2). This approach implies a dissociation between the definition of the combination and its actual implementation, which is common to all referee-based combinations. As a result, RefereeToolbox is designed with the aim to be generic and evolutive. It is composed of three distinct classes of objects:

- A class for the logical representation of the information; this class is generic and incremental; are yet implemented structures as:
  - Powerset,
  - Free Boolean algebra,
  - Superpowerset,
  - Open/closed hyperpowerset,
  - $\cdots \quad \longrightarrow$ make yours!
- A class for defining the referee functions; this class is generic and incremental; are yet defined referee functions for:
  - Dempster-Shafer combination,
  - Disjunctive combination,
  - Dubois&Prade combination,
  - PCR6 combination,
  - PCR# combination,
  - $\cdots \quad \longrightarrow$ make yours!
- A class defining the Basic Belief Assignment, and related processes; this class is generic and incremental; two referee-based approaches are implemented for computing combinations:
  - An exact approach associated with a relaxation of the focal elements, so as to handle the combinatorics,

– A sampling-based approach.

The generic implementation of RefereeToolbox makes possible to combine these three classes and their instances without restriction.

2 What is a referee function?

For an introduction to the theory of referee function, please refer to [3].

2.1 Basic Belief Assignment

Let G^Θ be a *complete distributive lattice*. A *basic belief assignment* [1, 2] is a mapping m from elements of G^Θ onto $[0,1]$ such that:

$$\sum_{X \in G^\Theta} m(X) = 1 \quad \text{and} \quad m(\emptyset) = 0 \; . \tag{1}$$

2.2 Referee function

Definition. A referee function over G^Θ for s sources of information and with context γ is a mapping $X, Y_{1:s} \mapsto F(X|Y_{1:s};\gamma)$ defined on propositions $X, Y_{1:s} \in G^\Theta$, which satisfies for any $X, Y_{1:s} \in G^\Theta$:

$$F(X|Y_{1:s};\gamma) \geq 0 \;\text{ and }\; \sum_{X \in G^\Theta} F(X|Y_{1:s};\gamma) = 1 \; ,$$

A referee function for s sources of information is also called a s-ary referee function. The quantity $F(X|Y_{1:s};\gamma)$ is called a *conditional arbitrament* between $Y_{1:s}$ in favor of X. Notice that X is not necessary one of the propositions $Y_{1:s}$; typically, it could be a combination of them. The case $X = \emptyset$ is called the *rejection case*.

Fusion rule. Let be given s basic belief assignments (bba) $m_{1:s}$ and a s-ary referee function F with context $m_{1:s}$. Then, the fused bba $m_1 \oplus \cdots \oplus m_s[F] \stackrel{\Delta}{=} \oplus[m_{1:s}|F]$ based on the referee F is constructed as follows:

$$\begin{aligned} \oplus[m_{1:s}|F](X) &= \frac{I[X \neq \emptyset]}{1-z} \sum_{Y_{1:s} \in G^\Theta} F(X|Y_{1:s};m_{1:s}) \prod_{i=1}^{s} m_i(Y_i) \; , \\ \text{where } z &= \sum_{Y_{1:s} \in G^\Theta} F(\emptyset|Y_{1:s};m_{1:s}) \prod_{i=1}^{s} m_i(Y_i) \; , \\ & I[X \neq \emptyset] = 1 \text{ if } X \neq \emptyset, \text{ and } I[X \neq \emptyset] = 0 \text{ if } X = \emptyset. \end{aligned} \tag{2}$$

The value z is called the *rejection rate*.

2.3 Examples of referee functions

Dempster-shafer rule. Let be given s sources of information characterized by their bbas $m_{1:s}$. The fused bba m_{DST} obtained from $m_{1:s}$ by means of *Dempster-Shafer* fusion rule [1, 2]

is defined by:

$$\begin{cases} m_{\text{DST}}(\emptyset) = 0 \ , \\ m_{\text{DST}}(X) = \dfrac{m_{\text{c}}(X)}{1 - m_{\text{c}}(\emptyset)} \quad \text{for any } X \in G^{\Theta} \setminus \{\emptyset\} \ , \end{cases}$$

where $m_{\text{c}}(\cdot)$ corresponds to the conjunctive consensus:

$$m_{\text{c}}(X) \triangleq \sum_{\substack{Y_1 \cap \cdots \cap Y_s = X \\ Y_1, \cdots, Y_s \in G^{\Theta}}} \prod_{i=1}^{s} m_i(Y_i) \ .$$

Definition by referee function. The definition of a referee function for Dempster-Shafer is immediate:

$$m_{\text{DST}} = \oplus[m_{1:s}|F_{\text{DST}}] \ ,$$

$$\text{where } F_{\text{DST}}(X|Y_{1:s}; m_{1:s}) = I\left[X = \bigcap_{k=1}^{s} Y_k\right] \ .$$

References

[1] Dempster A.P., *Upper and Lower probabilities induced by a multivalued mapping*, Annals of Mathematical Statistics, vol. 83, pp. 325–339, 1967.

[2] Shafer G., *A mathematical theory of evidence*, Princeton University Press, 1976.

[3] Frédéric Dambreville, *Chap. 6: Definition of evidence fusion rules based on referee functions*, in Smarandache F. & Dezert J., Editors, *Applications and Advances on DSmT for Information Fusion (Collected Works)*, Vol. 3, American Research Press, 2009.

3 Code

Source Code: ArrayBoolean.java

```
1  /*
2   *   ArrayBoolean.java : part of package RefereeToolbox; Implementation of
3   *          Boolean algebra.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox.  If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of a Boolean structure by means of an array of {@code long}.
30  * The AND and OR operators are inherited from the operators {@code & } and {@code|
        | }
31  * working on {@code long}.
32  * <i>zero</i> is defined by zeroing all bits. <i>one</i> is defined by setting to
33  * 1 all active bits; the right bits of the {@code long } vector may be inactive,
34  * depending on the size of the Lattice.
35  * The complement/cocomplement operators are inherited from the operators {@code ~|
        }
36  *  working on {@code long}; exceeding bits are masked by a AND with <i>one</i>.
37  *
38  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric |
        Dambreville</A>
39  *
40  *
41  *
42  * <BR><BR><table border='1' cellPadding='4'><tr><td>
43  * <font color="#008000" style="font-family: georgia">
44  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
45  * <BR>
46  *   RefereeToolbox is free software: you can redistribute it and/or modify
47  *   it under the terms of the GNU General Public License as published by
48  *   the Free Software Foundation, either version 3 of the License, or
49  *   (at your option) any later version.<BR>
50  * <BR>
51  *   RefereeToolbox is distributed in the hope that it will be useful,
52  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
53  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
54  *   GNU General Public License for more details.<BR>
55  * <BR>
56  *   You should have received a copy of the GNU General Public License
57  *   along with RefereeToolbox. If not, see
58  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
59  * </font></table><BR>
60  *
61  */
62 public class ArrayBoolean<L extends ArrayBoolean<L>>
63         extends ArrayLattice<L> implements ComplementedLattice<L> {
64
```

```
65
66      ////////////////////////////////////////////////:
67      // public part
68      ///////////////
69
70      /**
71       * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
            within
72       * <i>this</i>.
73       * The complement/cocomplement operators are inherited from the operators {↩
            @code ~ }
74       * working on {@code long}; exceeding bits are masked by a AND with <i>one</i↩
            >.
75       * The complement and cocomplement are the same for {@link ArrayBoolean }
76       * <BR><BR>
77       * <b>Documentation inherited from {@link ComplementedLattice }:</b><BR>
78       * {@inheritDoc}
79       *
80       */
81      public L complement(L aProposition) {
82          long[] rightmem=aProposition._memory;
83          int i;
84          for(i=0;i<=size_mem_1;i++)  _memory[i]=~rightmem[i];
85          _memory[size_mem_1]&=highest_long_one;
86          return (L) this;
87      }
88
89      /**
90       * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
            within
91       * <i>this</i>.
92       * The complement/cocomplement operators are inherited from the operators {↩
            @code ~ }
93       * working on {@code long}; exceeding bits are masked by a AND with <i>one</i↩
            >.
94       * The complement and cocomplement are the same for {@link ArrayBoolean }
95       * <BR><BR>
96       * <b>Documentation inherited from {@link ComplementedLattice }:</b><BR>
97       * {@inheritDoc}
98       *
99       */
100     public L complement() {
101         int i;
102         for(i=0;i<=size_mem_1;i++)  _memory[i]=~_memory[i];
103         _memory[size_mem_1]&=highest_long_one;
104         return (L) this;
105     }
106
107     /**
108      * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
            within
109      * <i>this</i>.
110      * The complement/cocomplement operators are inherited from the operators {↩
            @code ~ }
111      * working on {@code long}; exceeding bits are masked by a AND with <i>one</i↩
            >.
112      * The complement and cocomplement are the same for {@link ArrayBoolean }
113      * <BR><BR>
114      * <b>Documentation inherited from {@link ComplementedLattice }:</b><BR>
115      * {@inheritDoc}
116      *
117      */
118     public L cocomplement(L aProposition) {
119         return complement(aProposition);
120     }
121
122     /**
123      * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
            within
124      * <i>this</i>.
125      * The complement/cocomplement operators are inherited from the operators {↩
            @code ~ }
126      * working on {@code long}; exceeding bits are masked by a AND with <i>one</i↩
            >.
```

```
127         * The complement and cocomplement are the same for {@link ArrayBoolean }
128         * <BR><BR>
129         * <b>Documentation inherited from {@link ComplementedLattice }:</b><BR>
130         * {@inheritDoc}
131         *
132         */
133        public L cocomplement() {
134            return complement();
135        }
136
137    }
```

Source Code: ArrayLattice.java

```
1  /*
2   *    ArrayLattice.java : part of package RefereeToolbox; Implementation of
3   *          Lattice.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of a Lattice structure by means of an array of {@code long}.
30  * The AND and OR operators are inherited from the operators {@code & } and {@code↩
        | }
31  * working on {@code long}.
32  * <i>zero</i> is defined by zeroing all bits. <i>one</i> is defined by setting to
33  * 1 all active bits; the right bits of the {@code long } vector may be inactive,
34  * depending on the size of the Lattice.
35  *
36  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
37  *
38  *
39  *
40  * <BR><BR><table border='1' cellPadding='4'><tr><td>
41  * <font color="#008000" style="font-family: georgia">
42  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
43  * <BR>
44  *    RefereeToolbox is free software: you can redistribute it and/or modify
45  *    it under the terms of the GNU General Public License as published by
46  *    the Free Software Foundation, either version 3 of the License, or
47  *    (at your option) any later version.<BR>
48  * <BR>
49  *    RefereeToolbox is distributed in the hope that it will be useful,
50  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
51  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
52  *    GNU General Public License for more details.<BR>
53  * <BR>
54  *    You should have received a copy of the GNU General Public License
55  *    along with RefereeToolbox. If not, see
56  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
57  * </font></table><BR>
58  *
59  */
60 public class ArrayLattice<L extends ArrayLattice<L>>
61         extends LatticeCommon<L> {
62
63     protected long[] _memory=null;
64
65     protected int size_mem_1=-1; // undefined, by default
66     protected long highest_long_one;
67
68
69     ////////////////////////////////////////////////:
70     // public part
```

```
71      ///////////////
72
73  /**
74   * Return a representation of the state of <i>this</i> printed as a String. States
75   * of this class are printed as the concatenation of the Hexadecimal ↩
       representation
76   * of the <i>long</i> array.
77   *
78   */
79      @Override
80      public String state() {
81          String theState="";
82          String tmp;
83          int i;
84          for(i=size_mem_1;i>=0;i--) {
85              tmp=Long.toHexString(_memory[i]);
86              theState+="x"+"0000000000000000".substring(0, 16-tmp.length())+tmp;
87          }
88          return theState;
89      }
90
91  /**
92   * Set <i>this</i> to zero, the neutral element for
93   * {@link Lattice#or(java.lang.Object, java.lang.Object) }.
94   * N.B. It is necessary to call {@link Lattice#size } or
95   * {@link Lattice#size(java.lang.Object) } before any first use of this
96   * method.
97   *
98   */
99      @Override
100     public L zero() {
101         int i;
102         for(i=0;i<=size_mem_1;i++) _memory[i]=0;
103         return (L) this;
104     }
105
106 /**
107  * Set <i>this</i> to one, the neutral element for
108  * {@link Lattice#and(java.lang.Object, java.lang.Object) }.
109  * N.B. It is necessary to call {@link Lattice#size } or
110  * {@link Lattice#size(java.lang.Object) } before any first use of this method.
111  *
112  */
113     @Override
114     public L one() {
115         int i;
116         for(i=0;i<size_mem_1;i++)  _memory[i]=0xFFFFFFFFFFFFFFFFL;
117         _memory[i]=highest_long_one;
118         return (L) this;
119     }
120
121 /**
122  * Compute the AND of <i>left</i> and <i>right</i> and store the result within <i>↩
       this</i>.
123  *
124  */
125     @Override
126     public L and(L left, L right) {
127         long[] leftmem=left._memory;
128         long[] rightmem=right._memory;
129         int i;
130         for(i=0;i<=size_mem_1;i++)  _memory[i]=leftmem[i]&rightmem[i];
131         return (L) this;
132     }
133
134 /**
135  * Compute the OR of <i>left</i> and <i>right</i> and store the result within <i>↩
       this</i>.
136  *
137  */
138     @Override
139     public L or(L left, L right) {
140         long[] leftmem=left._memory;
141         long[] rightmem=right._memory;
```

```
142         int i;
143         for(i=0;i<=size_mem_1;i++)  _memory[i]=leftmem[i]|rightmem[i];
144         return (L) this;
145     }
146 
147 /**
148  * Answer <i>true</i> if the proposition stored within <i>this</i> intersects the
149  * proposition contained within <i>aProposition</i> (<i>i.e.</i> the intersection
150  * of both propositions is not <i>zero</i>). Answer <i>false</i> otherwise.
151  *
152  */
153     @Override
154     public  boolean intersects(L aProposition) {
155         long[] rightmem=aProposition._memory;
156         int i;
157         for(i=0;i<=size_mem_1;i++) {
158             if((_memory[i]&rightmem[i])!=0) return true;
159         }
160         return false;
161     }
162 
163 /**
164  * Answer <i>true</i> if the proposition stored within <i>this</i> contains the
165  * proposition contained within <i>aProposition</i>. Answer <i>false</i> otherwise↩
        .
166  *
167  */
168     @Override
169     public boolean contains(L aProposition) {
170         long[] rightmem=aProposition._memory;
171         int i;
172         for(i=0;i<=size_mem_1;i++) {
173             if((_memory[i]&rightmem[i])!=rightmem[i]) return false;
174         }
175         return true;
176     }
177 
178 /**
179  * Compare <i>this</i> to <i>aProposition</i> and answer negative integer, zero,
180  * positive integer as <i>this</i> is less than, equal to, or greater than
181  * <i>aProposition</i>.
182  * This comparison is related to a total ordering of the proposition; <em>it is ↩
        not
183  * related to the partial order implied by the logical operators AND and OR.</em>
184  *
185  */
186     @Override
187     public int compareTo(L aProposition) {
188         long[] rightmem=aProposition._memory;
189         int i;
190         long delta;
191         for(i=0;i<=size_mem_1;i++) {
192             delta=_memory[i]-rightmem[i];
193             if(delta>0) return 1;
194             if(delta<0) return -1;
195         }
196         return 0;
197     }
198 
199 /**
200  * Resize <i>this</i> at the same size than <i>input</i>.
201  *
202  */
203     @Override
204     public L size(L input) {
205         size_mem_1=input.size_mem_1;
206         highest_long_one=input.highest_long_one;
207         _memory=new long[size_mem_1+1];
208         return (L) this;
209     }
210 
211 
212  /**
213  * Do an exact copy of input into <i>this</i>.
```

```
214  *
215  */
216      @Override
217      public L duplicate(L input) {
218          size(input);
219          int i;
220          for(i=0;i<=size_mem_1;i++) _memory[i]=input._memory[i];
221          return (L) this;
222      }
223
224  }
```

Source Code: Assignment.java

```
1  /*
2   *    Assignment.java : part of package RefereeToolbox; Class encoding an
3   *          assignment of a basic belief to a proposition.
4   ************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Container instrumental to all classes implementing {@link minAssignment } and
30  * to class RefereeFunctionDefault and its subclasses;
31  * a structure encoding an assignment of a basic belief to a proposition.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *    RefereeToolbox is free software: you can redistribute it and/or modify
42  *    it under the terms of the GNU General Public License as published by
43  *    the Free Software Foundation, either version 3 of the License, or
44  *    (at your option) any later version.<BR>
45  * <BR>
46  *    RefereeToolbox is distributed in the hope that it will be useful,
47  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *    GNU General Public License for more details.<BR>
50  * <BR>
51  *    You should have received a copy of the GNU General Public License
52  *    along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57     public class Assignment<Prop> {
58 /**
59  * This subfield stores the proposition of the assignment typed {@link Assignment ↩
        }.
60  *
61  */
62         public Prop attribute;
63 /**
64  * This subfield stores the value of the assignment.
65  *
66  */
67         public Double value;
68     }
```

Source Code: BasicBeliefAssignment.java

```
1  /*
2   *    BasicBeliefAssignment.java : part of package RefereeToolbox; Interface for
3   *          Basic Belief Assigments.
4   ************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Contains the minimal declaration for classes managing belief assignment
32  * structures, belief assignment processes and fusers.
33  * This interface is implemented by all the these classes.
34  *
35  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
36  *
37  *
38  *
39  * <BR><BR><table border='1' cellPadding='4'><tr><td>
40  * <font color="#008000" style="font-family: georgia">
41  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
42  * <BR>
43  *    RefereeToolbox is free software: you can redistribute it and/or modify
44  *    it under the terms of the GNU General Public License as published by
45  *    the Free Software Foundation, either version 3 of the License, or
46  *    (at your option) any later version.<BR>
47  * <BR>
48  *    RefereeToolbox is distributed in the hope that it will be useful,
49  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
50  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
51  *    GNU General Public License for more details.<BR>
52  * <BR>
53  *    You should have received a copy of the GNU General Public License
54  *    along with RefereeToolbox. If not, see
55  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
56  * </font></table><BR>
57  *
58  */
59 public interface BasicBeliefAssignment<Prop extends Lattice<Prop>,
60                        B extends BasicBeliefAssignment<Prop,B> >
61                                    extends minAssignment<Prop> , Cloneable {
62
63 /**
64  * Clear the assignments stored within {@link BasicBeliefAssignment }, and
65  * store new assignments from the collection of assignment, <i>anAssignmentTab</i↩
      >.
66  *
67  */
68     boolean load(Collection<Assignment<Prop>> anAssignmentTab);
69
70 /**
```

```
71  * Do an exact copy of input into <i>this</i>.
72  *
73  */
74     B duplicate(B input); // make this a duplicate of input
75
76 /**
77  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
78  * the Class of <i>this</i>).
79  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
80  * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
81  * the non generic (typically final) sub-classes</u>.
82  * <br><br>
83  * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
84  * definition of {@link BasicBeliefAssignment#instance() } is as follows:
85  * <br><br>
86  * <font color="#004488"><code>
87  * {@code @Override}
88  * <br>
89  * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
90  * </code></font>
91  *
92  */
93     B instance();
94
95 /**
96  * Create  a clone of <i>this</i>.
97  *
98  */
99     B clone();
100
101 /**
102  * Make a mix of the basic belief assignments stored within the entry, <i>bbaIn</i>
103  * according to their respective weight <i>weight</i>. This mix is stored into <i>this</i>.
104  *
105  */
106     B mix(ArrayList<B> bbaIn, double[] weight);
107
108 /**
109  * Called each time a change is done to <i>this</i>. This is usefull for processes which need an
110  * initialization after each change of the assignments. Typically, the
111  * sampling methods require an action of {@link BasicBeliefAssignment#update_notification() }
112  * in order to decide for a new computation of the sampling tables, which are made
113  * necessary by a change of the assignments.
114  *
115  */
116     void update_notification(); //
117
118 /**
119  * Return a representation of the state of <i>this</i> printed as
120  * a String. Typically, this state is the list of all stored assignments (proposition
121  * and value). Depending on the parameter <i>choix</i> in entry, the representation
122  * is modified as folllows:<br><br>
123  * <i>choix==0</i>  --  Nothing is printed.
124  * <br>
125  *<i>choix==1</i>  --  Assignments are printed in increasing order of their
126  * propositions,
127  * <br>
128  *<i>choix==2</i>  --  Assignments are printed in decreasing order of their
129  * value,
130  * <br>
131  *<i>choix==3</i>  --  Do both print in that order.
132  * <br>
133  *
134  */
135     String state(int choix); // a string representing the state of the BBA
136
137 /**
138  * Remove all assignments from <i>this</i>. The class instance is cleared.
```

```
139  *
140  */
141     void clear();
142
143 /**
144  * Add an assignment characterized by a proposition <i>aProposition</i> and a ↩
         value
145  * <i>anAssignmentValue</i> to <i>this</i>.
146  * If an assignment already exists for <i>aProposition</i>, say with value <i>↩
         theOldValue</i>,
147  * then the new assignement of <i>aProposition</i> is stored with value
148  * <i>theOldValue+anAssignmentValue</i>.
149  *
150  */
151     boolean add(Prop aProposition, double anAssignmentValue);
152
153 /**
154  * Do exactly as {@link BasicBeliefAssignment#add(RefereeToolbox.Lattice, double) ↩
         }
155  * but takes an entry of type {@link Assignment }, which is actually
156  * equivalent.
157  *
158  */
159     boolean add(Assignment<Prop> anAssignment);
160
161 /**
162  * Add all assignments of collection <i>assignTab</i> to <i>this</i>.
163  * This method is equivalent to applying
164  * {@link BasicBeliefAssignment#add(RefereeToolbox.Assignment) }
165  * to each assignment of collection <i>assignTab</i>.
166  *
167  */
168     boolean addAll(Collection<Assignment<Prop>> assignTab);
169
170 /**
171  * Remove the assignment related to proposition <i>aProposition</i>, if there is
172  * such assignment stored within <i>this</i>.
173  * Return the value assigned to the proposition.
174  *
175  */
176     double remove(Prop aProposition);
177 }
```

Source Code: BBACommon.java

```
1  /*
2   *   BBACommon.java : part of package RefereeToolbox; Basical implementation of
3   *         the Basic Belief Assigment.
4   *****************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  *****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Basical implementation of the Basic Belief Assigment structure.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public class BBACommon <Prop extends Lattice<Prop>,
58                                             B extends BBACommon<Prop,B> >
59                                             implements BasicBeliefAssignment<Prop,↩
                                                B> {
60
61
62     ////////////////////////////////////////////////:
63     // public part
64     ///////////////
65
66     /**
67      * This method is not defined for class {@link BBACommon } and generates an ↩
            error
68      * report and a code exit.
69      * <BR><BR>
```

```
70          * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
71          * {@inheritDoc}
72          */
73         public B duplicate(B input) {
74             System.err.println("Error :: "+this.getClass().getName() +
75                     ".duplicate(B) is not implemented!");
76             System.err.println("Please implement clone() in your class with @Override"↩
                    );
77             System.exit(0);
78             return null;
79         }
80 
81         @Override
82         public B clone() {
83             return instance().duplicate((B) this);
84         }
85 
86         /**
87          * This method is not defined for class {@link BBACommon } and generates an ↩
                error
88          * report and a code exit.
89          * <BR><BR>
90          * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
91          * {@inheritDoc}
92          */
93         public B instance() {
94             System.err.println("Error :: "+this.getClass().getName() +
95                     ".clone() is not implemented!");
96             System.err.println("Please implement clone() in your class with @Override"↩
                    );
97             System.exit(0);
98             return null;
99         }
100 
101         /**
102          * This method is not defined for class {@link BBACommon } and produces the ↩
                String
103          * {@code "UNDEFINED" }.
104          * <BR><BR>
105          * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
106          * {@inheritDoc}
107          */
108         public String state(int choix) {
109             return "UNDEFINED";
110         }
111 
112         /**
113          * This method is not defined for class {@link BBACommon } and generates an ↩
                error
114          * report and a code exit.
115          * <BR><BR>
116          * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
117          * {@inheritDoc}
118          */
119         public ArrayList<Assignment<Prop>> toArray() {
120             System.err.println("Error :: "+this.getClass().getName() +
121                     ".toArray() is not implemented!");
122             System.err.println("Please implement toArray() in your class with ↩
                    @Override");
123             System.exit(0);
124             return null;
125         }
126 
127         public boolean load(Collection<Assignment<Prop>> anAssignmentTab) {
128             clear();
129             addAll(anAssignmentTab);
130             return true;
131         }
132 
133         /**
134          * This method is not defined for class {@link BBACommon } and generates an ↩
                error
135          * report and a code exit.
136          * <BR><BR>
```

```
137         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
138         * {@inheritDoc}
139         */
140        public void clear() {
141            System.err.println("Error :: "+this.getClass().getName() +
142                    ".clear() is not implemented!");
143            System.err.println("Please implement clear() in your class with @Override"↩
                    );
144            System.exit(0);
145        }
146
147        /**
148         * This method is not defined for class {@link BBACommon } and generates an ↩
                error
149         * report and a code exit.
150         * <BR><BR>
151         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
152         * {@inheritDoc}
153         */
154        public boolean add(Prop aProposition, double anAssignmentValue) {
155            System.err.println("Error :: "+this.getClass().getName() +
156                    ".add() is not implemented!");
157            System.err.println("Please implement add() in your class with @Override");
158            System.exit(0);
159            return true;
160        }
161
162        public boolean add(Assignment<Prop> anAssignment) {
163            return add(anAssignment.attribute,anAssignment.value);
164        }
165
166        public boolean addAll(Collection<Assignment<Prop>> assignTab) {
167            //bbaTree;
168            boolean test=true;
169            for (Iterator<Assignment<Prop>> it = assignTab.iterator();
170                                                    it.hasNext(); ) {
171                test=test&&add(it.next());
172           }
173            return test;
174        }
175
176        /**
177         * This method is not defined for class {@link BBACommon } and generates an ↩
                error
178         * report and a code exit.
179         * <BR><BR>
180         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
181         * {@inheritDoc}
182         */
183        public Assignment<Prop> findProposition(Assignment<Prop> anAssignment) {
184            System.err.println("Error :: "+this.getClass().getName() +
185                    ".findProposition(Assignment<Prop>) is not implemented!");
186            System.err.println("Please implement findProposition(Assignment<Prop>)" +
187                    " in your class with @Override");
188            System.exit(0);
189            return null;
190        }
191
192        public Assignment<Prop> findProposition(Prop aProposition) {
193            Assignment<Prop> searchAssign = new Assignment<Prop>();
194            searchAssign.attribute=aProposition;
195            return findProposition(searchAssign);
196        }
197
198        public double m(Prop aProposition) {
199
200            Assignment<Prop>  foundPair=findProposition(aProposition);
201            if(foundPair!=null) {
202                    return foundPair.value;
203                }
204            return 0.;
205        }
206
207        /**
```

```
208         * This method is not defined for class {@link BBACommon } and generates an error
209         * report and a code exit.
210         * <BR><BR>
211         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
212         * {@inheritDoc}
213         */
214        public double Bel(Prop aProposition) {
215            System.err.println("Error :: "+this.getClass().getName() +
216                    ".Bel(Prop) is not implemented!");
217            System.err.println("Please implement Bel(Prop) in your class with @Override");
218            System.exit(0);
219            return -1;
220        }
221
222        /**
223         * This method is not defined for class {@link BBACommon } and generates an error
224         * report and a code exit.
225         * <BR><BR>
226         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
227         * {@inheritDoc}
228         */
229        public double Pl(Prop aProposition) {
230            System.err.println("Error :: "+this.getClass().getName() +
231                    ".Pl(Prop) is not implemented!");
232            System.err.println("Please implement Pl(Prop) in your class with @Override");
233            System.exit(0);
234            return -1;
235        }
236
237        /**
238         * This method is not defined for class {@link BBACommon } and generates an error
239         * report and a code exit.
240         * <BR><BR>
241         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
242         * {@inheritDoc}
243         */
244        public double remove(Prop aProposition) {
245            System.err.println("Error :: "+this.getClass().getName() +
246                    ".remove(Prop) is not implemented!");
247            System.err.println("Please implement remove(Prop) in your class with @Override");
248            System.exit(0);
249            return -1.;
250        }
251
252        /**
253         * This method is not defined for class {@link BBACommon } and generates an error
254         * report and a code exit.
255         * <BR><BR>
256         * <b>Documentation inherited from {@link BasicBeliefAssignment }:</b><BR>
257         * {@inheritDoc}
258         */
259        public B mix(ArrayList<B> bbaIn, double[] weight) {
260            System.err.println("Error :: "+this.getClass().getName() +
261                    ".mix(ArrayList<B>, double[]) is not implemented!");
262            System.err.println("Please implement mix(ArrayList<B>, double[]) in your" +
263                    " class with @Override");
264            System.exit(0);
265            return null;
266        }
267
268        public void update_notification() { // called each time a change is done
269        }
270
271    }
```

Source Code: BBAFuser.java

```
1  /*
2   *   BBAFuser.java : part of package RefereeToolbox; Interface for the fusion of
3   *          Basic Belief Assigments.
4   ************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Methods for fusing Basic Belief Assignments. This interface concerns direct
32  * rule implementations. It does not concern referee-based fusers, for which there
33  * are dedicated interfaces {@link BBARefereeFuser}, {@link SampledBBARefereeFuser}.
34  *
35  * @see          BBARefereeFuser
36  * @see          SampledBBARefereeFuser
37  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric Dambreville</A>
38  *
39  *
40  *
41  * <BR><BR><table border='1' cellPadding='4'><tr><td>
42  * <font color="#008000" style="font-family: georgia">
43  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
44  * <BR>
45  *   RefereeToolbox is free software: you can redistribute it and/or modify
46  *   it under the terms of the GNU General Public License as published by
47  *   the Free Software Foundation, either version 3 of the License, or
48  *   (at your option) any later version.<BR>
49  * <BR>
50  *   RefereeToolbox is distributed in the hope that it will be useful,
51  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
52  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
53  *   GNU General Public License for more details.<BR>
54  * <BR>
55  *   You should have received a copy of the GNU General Public License
56  *   along with RefereeToolbox. If not, see
57  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
58  * </font></table><BR>
59  *
60  */
61 public interface BBAFuser<Prop extends Lattice<Prop>, B extends BBAFuser<Prop,B> >
62                                         extends BasicBeliefAssignment<Prop,B> {
63
64 /**
65  * Compute the combination (fusion) of basic belief assignments <i>left</i> and
66  * <i>right</i> and store the result within <i>this</i>.
67  *
68  */
69     B fuse(B left, B right);
70
```

```
71 /**
72  * Compute the combination (fusion) of the basic belief assignments within array
73  * <i>bbaIn</i> and store the result within <i>this</i>.
74  *
75  */
76     B fuse(ArrayList<B> bbaIn);
77 
78 /**
79  * Return the conflict of the last combination.
80  *
81  */
82     B conflict();
83 
84 }
```

Source Code: BBARefereeFuser.java

```
1  /*
2   *   BBARefereeFuser.java : part of package RefereeToolbox; Interface for fuser
3   *           based on referee function.
4   ****************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Methods for fusing Basic Belief Assignments by the means of referee functions
32  * and on the basis of an exact computation. This interface does not concern ↩
        direct
33  * rule implementations, {@link BBAFuser}, or implementations based on referee
34  * sampling, {@link SampledBBARefereeFuser}.
35  *
36  * @see          BBAFuser
37  * @see          SampledBBARefereeFuser
38  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
39  *
40  *
41  *
42  * <BR><BR><table border='1' cellPadding='4'><tr><td>
43  * <font color="#008000" style="font-family: georgia">
44  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
45  * <BR>
46  *   RefereeToolbox is free software: you can redistribute it and/or modify
47  *   it under the terms of the GNU General Public License as published by
48  *   the Free Software Foundation, either version 3 of the License, or
49  *   (at your option) any later version.<BR>
50  * <BR>
51  *   RefereeToolbox is distributed in the hope that it will be useful,
52  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
53  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
54  *   GNU General Public License for more details.<BR>
55  * <BR>
56  *   You should have received a copy of the GNU General Public License
57  *   along with RefereeToolbox. If not, see
58  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
59  * </font></table><BR>
60  *
61  */
62 public interface BBARefereeFuser<Prop extends Lattice<Prop>, B extends ↩
        BBARefereeFuser<Prop,B>>
63                                         extends BasicBeliefAssignment<Prop,B> {
64
65 /**
66  * Compute the combination (fusion) of basic belief assignments <i>left</i> and
67  * <i>right</i> by means of the referee function <i>theRefereeFunction</i> and
68  * store the result within <i>this</i>.
69  *
```

```
70   */
71      B fuse(B left, B right, RefereeFunctionDefault<Prop> theRefereeFunction);
72
73  /**
74   * Compute the combination (fusion) of basic belief assignments within array
75   * <i>bbaIn</i> by means of the referee function <i>theRefereeFunction</i> and
76   * store the result within <i>this</i>.
77   *
78   */
79      B fuse(ArrayList<B> bbaIn, RefereeFunctionDefault<Prop> theRefereeFunction);
80
81  /**
82   * Return the conflict of the last combination.
83   *
84   */
85      double conflict();
86
87  }
```

Source Code: Closedhyperpowerset.java

```
1  /*
2   *    Closedhyperpowerset.java : part of package RefereeToolbox; Implementation
3   *          of closed hyperpowerset.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of a closed hyperpowerset structure. A closed hyperpowerset is
30  * an hyperpowerset defined on the basis of a superpowerset: it is generated
31  * from the atomic propositions of the superpowerset but without the use of
32  * the complement operator of the superpowerset.
33  * <BR><BR>
34  * The closed hyperpowerset implies a closed world hypothesis, that is:
35  *<BR><BR>
36  * <i> <b>OR</b><sub>0 =< i < sizeFrame</sub> atomic(i)  ==  one</i>
37  * <BR><BR>
38  * Although generated without the complement of the superpowerset, the closed
39  * hyperpowerset actually has proper complement and cocomplement operators.
40  * The  complement and cocomplement are distinct operators for the closed ↩
        hyperpowerset.
41  *
42  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
43  *
44  *
45  *
46  * <BR><BR><table border='1' cellPadding='4'><tr><td>
47  * <font color="#008000" style="font-family: georgia">
48  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
49  * <BR>
50  *    RefereeToolbox is free software: you can redistribute it and/or modify
51  *    it under the terms of the GNU General Public License as published by
52  *    the Free Software Foundation, either version 3 of the License, or
53  *    (at your option) any later version.<BR>
54  * <BR>
55  *    RefereeToolbox is distributed in the hope that it will be useful,
56  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
57  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
58  *    GNU General Public License for more details.<BR>
59  * <BR>
60  *    You should have received a copy of the GNU General Public License
61  *    along with RefereeToolbox. If not, see
62  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
63  * </font></table><BR>
64  *
65  */
66 public class Closedhyperpowerset<L extends Closedhyperpowerset<L>> extends ↩
        Superpowerset<L> {
67     protected L theZero = null;
68     protected L theOne = null;
69
```

```
70      /**
71       * Compute the <b>complement</b> of <i>aProposition</i> and store the result
72       * within <i>this</i>.
73       * The complement operator for the closed hyperpowerset is defined by:
74       * <BR><BR>
75       * <i> complement(zero) = one</i>
76       * <BR>
77       * <i> complement(X) = zero</i> if <i>X != zero</i>
78       *
79       */
80      @Override
81      public L complement(L aProposition) {
82          if(theZero==null) {
83              theZero = instanceNsize().zero();
84              theOne = instanceNsize().one();
85          }
86          if(aProposition.compareTo(theZero)==0) one();
87          else zero();
88          return (L) this;
89      }
90
91      /**
92       * Compute the <b>complement</b> of <i>this</i> and store the result within
93       * <i>this</i>.
94       * The complement operator for the closed hyperpowerset is defined by:
95       * <BR><BR>
96       * <i> complement(zero) = one</i>
97       * <BR>
98       * <i> complement(X) = zero</i> if <i>X != zero</i>
99       *
100      */
101     @Override
102     public L complement() {
103         if(theZero==null) {
104             theZero = instanceNsize().zero();
105             theOne = instanceNsize().one();
106         }
107         if(compareTo(theZero)==0) one();
108         else zero();
109         return (L) this;
110     }
111
112     /**
113      * Compute the <b>cocomplement</b> of <i>aProposition</i> and store the result
114      * within <i>this</i>.
115      * The cocomplement operator for the closed hyperpowerset is defined by:
116      * <BR><BR>
117      * <i> cocomplement(X) = <b>OR</b><sub>Z AND X != Z</sub> Z</i>
118      *
119      */
120     @Override
121     public L cocomplement(L aProposition) {
122         if(theZero==null) {
123             theZero = instanceNsize().zero();
124             theOne = instanceNsize().one();
125         }
126         L tmpProposition = instanceNsize();
127         L finalProposition = instanceNsize();
128         finalProposition.zero();
129         int i;
130         for(i=0;i<size(-1);i++) {
131             tmpProposition.atomic(i);
132             if(!aProposition.contains(tmpProposition))
133                         finalProposition.or(finalProposition, tmpProposition);
134         }
135         return duplicate(finalProposition);
136     }
137
138     /**
139      * Compute the <b>cocomplement</b> of <i>this</i> and store the result
140      * within <i>this</i>.
141      * The cocomplement operator for the closed hyperpowerset is defined by:
142      * <BR><BR>
143      * <i> cocomplement(X) = <b>OR</b><sub>Z AND X != Z</sub> Z</i>
```

```
144         *
145         */
146        @Override
147        public L cocomplement() {
148            if(theZero==null) {
149                theZero = instanceNsize().zero();
150                theOne = instanceNsize().one();
151            }
152            L tmpProposition = instanceNsize();
153            L finalProposition = instanceNsize();
154            finalProposition.zero();
155            int i;
156            for(i=0;i<size(-1);i++) {
157                tmpProposition.atomic(i);
158                if(!contains(tmpProposition))
159                            finalProposition.or(finalProposition, tmpProposition);
160            }
161            return duplicate(finalProposition);
162        }
163
164    }
```

Source Code: ComplementedLattice.java

```
1  /*
2   *    ComplementedLattice.java : part of package RefereeToolbox; Interface
3   *          for complemented Lattice.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Contains the methods for a class implementing a Lattice structure with a
30  * complement or a pseudo-complement operator.
31  * Boolean algebra, powerset, as well as Heyting algebra or closed/open ↩
        Hyperpowersets
32  * are examples of complemented lattices.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *    RefereeToolbox is free software: you can redistribute it and/or modify
43  *    it under the terms of the GNU General Public License as published by
44  *    the Free Software Foundation, either version 3 of the License, or
45  *    (at your option) any later version.<BR>
46  * <BR>
47  *    RefereeToolbox is distributed in the hope that it will be useful,
48  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *    GNU General Public License for more details.<BR>
51  * <BR>
52  *    You should have received a copy of the GNU General Public License
53  *    along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public interface ComplementedLattice<L> extends Lattice<L> {
59
60 /**
61  * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
        within
62  * <i>this</i>.
63  * <BR><BR>
64  * <b>Definition.</b> The complement of a proposition <i>X</i> of a lattice <i>L</i>↩
        i>
65  * is: <i> comp(X) = <b>OR</b><sub>Y : Y AND X = zero</sub> Y </i>
66  * <BR><BR>
67  * <b>Theorem 1.</b> If the lattice is a boolean algebra, then complement and ↩
        cocomplement
```

```
68   * are identical, <i>comp(X) =cocomp(X)</i>
69   * <BR>
70   * <b>Theorem 2.</b> <i>cocomp(X)</i> contains <i>comp(X)</i>
71   *
72   * @see ComplementedLattice#cocomplement(java.lang.Object)
73   */
74      L complement(L aProposition);
75
76  /**
77   * Compute the <b>complement</b> of <i>this</i> and store the result within
78   * <i>this</i>.
79   * <BR><BR>
80   * <b>Definition.</b> The complement of a proposition <i>X</i> of a lattice <i>L</i>
       i>
81   * is: <i> comp(X) = <b>OR</b><sub>Y : Y AND X = zero</sub> Y </i>
82   * <BR><BR>
83   * <b>Theorem 1.</b> If the lattice is a boolean algebra, then complement and
       cocomplement
84   * are identical, <i>comp(X) =cocomp(X)</i>
85   * <BR>
86   * <b>Theorem 2.</b> <i>cocomp(X)</i> contains <i>comp(X)</i>
87   *
88   * @see ComplementedLattice#cocomplement()
89   */
90      L complement();
91
92  /**
93   * Compute the <b>cocomplement</b> of <i>aProposition</i> and store the result
94   * within <i>this</i>.
95   *
96   * <BR><BR>
97   * <b>Definition.</b> The cocomplement of a proposition <i>X</i> of a lattice <i>L
       </i>
98   * is: <i> cocomp(X) = <b>AND</b><sub>Z : Z OR X = one</sub> Z </i>
99   * <BR><BR>
100  * <b>Theorem 1.</b> If the lattice is a boolean algebra, then complement and
       cocomplement
101  * are identical, <i>comp(X) =cocomp(X)</i>
102  * <BR>
103  * <b>Theorem 2.</b> <i>cocomp(X)</i> contains <i>comp(X)</i>
104  *
105  * @see ComplementedLattice#complement(java.lang.Object)
106  */
107     L cocomplement(L aProposition);
108
109 /**
110  * Compute the <b>cocomplement</b> of <i>this</i> and store the result
111  * within <i>this</i>.
112  *
113  * <BR><BR>
114  * <b>Definition.</b> The cocomplement of a proposition <i>X</i> of a lattice <i>L
       </i>
115  * is: <i> cocomp(X) = <b>AND</b><sub>Z : Z OR X = one</sub> Z </i>
116  * <BR><BR>
117  * <b>Theorem 1.</b> If the lattice is a boolean algebra, then complement and
       cocomplement
118  * are identical, <i>comp(X) =cocomp(X)</i>
119  * <BR>
120  * <b>Theorem 2.</b> <i>cocomp(X)</i> contains <i>comp(X)</i>
121  *
122  * @see ComplementedLattice#complement()
123  */
124     L cocomplement();
125 }
```

Source Code: finalClosedhyperpowerset.java

```
1  /*
2   *   finalClosedhyperpowerset.java : part of package RefereeToolbox;
3   *          Implementation of closed hyperpowerset.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Final implementation of class {@link Closedhyperpowerset }. This class is not ↩
        generic, and
30  * thus, allows and needs instantiation. The method {@link ↩
        finalClosedhyperpowerset#instance() }
31  * is implemented.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public final class finalClosedhyperpowerset  extends Closedhyperpowerset<↩
        finalClosedhyperpowerset> {
58
59
60      ////////////////////////////////////////////////:
61      // public part
62      ///////////////
63
64 /**
65  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
66  * the Class of <i>this</i>).
67  * N.B. {@link Lattice#instance() } cannot be defined
68  * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
```

```
69   * defined with the non generic (typically final) sub-classes</u>.
70   * <br><br>
71   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
72   * definition of {@link Lattice#instance() } is as follows:
73   * <br><br>
74   * <font color="#004488"><code>
75   * {@code @Override}
76   * <br>
77   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
78   * </code></font>
79   *
80   */
81      @Override
82      public finalClosedhyperpowerset instance() { return new ↩
            finalClosedhyperpowerset(); }
83  }
```

Source Code: finalFreeboolean.java

```
1  /*
2   *   finalFreeboolean.java : part of package RefereeToolbox; Implementation of
3   *          free Boolean algebra.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Final implementation of class {@link Freeboolean }. This class is not generic, ↩
        and
30  * thus, allows and needs instantiation. The method {@link finalFreeboolean#↩
        instance() }
31  * is implemented.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public final class finalFreeboolean extends Freeboolean<finalFreeboolean>  {
58
59
60      ////////////////////////////////////////////////:
61      // public part
62      ///////////////
63
64 /**
65  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
66  * the Class of <i>this</i>).
67  * N.B. {@link Lattice#instance() } cannot be defined
68  * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
69  * defined with the non generic (typically final) sub-classes</u>.
```

```
70   * <br><br>
71   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
72   * definition of {@link Lattice#instance() } is as follows:
73   * <br><br>
74   * <font color="#004488"><code>
75   * {@code @Override}
76   * <br>
77   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
78   * </code></font>
79   *
80   */
81      @Override
82      public finalFreeboolean instance() { return new finalFreeboolean(); }
83  }
```

Source Code: finalOpenhyperpowerset.java

```
1  /*
2   *   finalOpenhyperpowerset.java : part of package RefereeToolbox;
3   *          Implementation of open hyperpowerset.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Final implementation of class {@link Openhyperpowerset }. This class is not ↩
        generic, and
30  * thus, allows and needs instantiation. The method {@link finalOpenhyperpowerset#↩
        instance() }
31  * is implemented.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public final class finalOpenhyperpowerset  extends Openhyperpowerset<↩
        finalOpenhyperpowerset> {
58
59
60      ////////////////////////////////////////////////:
61      // public part
62      ///////////////
63
64 /**
65  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
66  * the Class of <i>this</i>).
67  * N.B. {@link Lattice#instance() } cannot be defined
68  * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
```

```
69   * defined with the non generic (typically final) sub-classes</u>.
70   * <br><br>
71   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
72   * definition of {@link Lattice#instance() } is as follows:
73   * <br><br>
74   * <font color="#004488"><code>
75   * {@code @Override}
76   * <br>
77   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
78   * </code></font>
79   *
80   */
81      @Override
82      public final0penhyperpowerset instance() { return new final0penhyperpowerset()↩
            ; }
83  }
```

Source Code: finalPowerset.java

```
1  /*
2   *   finalPowerset.java : part of package RefereeToolbox; Implementation of
3   *         powerset.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Final implementation of class {@link Powerset }. This class is not generic, and
30  * thus, allows and needs instantiation. The method {@link finalPowerset#instance↩
        () }
31  * is implemented.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public final class finalPowerset extends Powerset<finalPowerset> {
58
59
60     ////////////////////////////////////////////////:
61     // public part
62     ///////////////
63
64 /**
65  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
66  * the Class of <i>this</i>).
67  * N.B. {@link Lattice#instance() } cannot be defined
68  * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
69  * defined with the non generic (typically final) sub-classes</u>.
70  * <br><br>
```

```
71   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
72   * definition of {@link Lattice#instance() } is as follows:
73   * <br><br>
74   * <font color="#004488"><code>
75   * {@code @Override}
76   * <br>
77   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
78   * </code></font>
79   *
80   */
81      @Override
82      public finalPowerset instance() { return new finalPowerset(); }
83  }
```

Source Code: finalRefereeFuserRTS_Closedhyperpowerset.java

```
1  /*
2   *   finalRefereeFuserRTS_Closedhyperpowerset.java : part of package
3   *         RefereeToolbox; Implementation of fuser based on referee function.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeFuserRTS } specialized for
30  * Lattice structures typed {@link finalClosedhyperpowerset }.
31  * An instanciation method {@link finalRefereeFuserRTS_Closedhyperpowerset#↩
        instance() } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeFuserRTS_Closedhyperpowerset extends RefereeFuserRTS<↩
        finalClosedhyperpowerset,
59                                                                 finalRefereeFuserRTS_Closedhyperpowerset↩
            > {
60
61      ////////////////////////////////////////////////:
62      // public part
63      ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
```

```
 * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
 * the non generic (typically final) sub-classes</u>.
 * <br><br>
 * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
 * definition of {@link BasicBeliefAssignment#instance() } is as follows:
 * <br><br>
 * <font color="#004488"><code>
 * {@code @Override}
 * <br>
 * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
 * </code></font>
 *
 */
    @Override
    public finalRefereeFuserRTS_Closedhyperpowerset instance() { return new ↩
        finalRefereeFuserRTS_Closedhyperpowerset(); }

}
```

Source Code: finalRefereeFuserRTS_Freeboolean.java

```
1  /*
2   *   finalRefereeFuserRTS_Freeboolean.java : part of package RefereeToolbox;
3   *          Implementation of fuser based on referee function.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeFuserRTS } specialized for
30  * Lattice structures typed {@link finalFreeboolean }.
31  * An instanciation method {@link finalRefereeFuserRTS_Freeboolean#instance() } is
32  * implemented for this non generic class.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeFuserRTS_Freeboolean extends RefereeFuserRTS<↩
        finalFreeboolean,
59                                                                 finalRefereeFuserRTS_Freeboolean ↩
                                                                       > {
60
61      ////////////////////////////////////////////////:
62      // public part
63      ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
69  * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
```

```
70   * the non generic (typically final) sub-classes</u>.
71   * <br><br>
72   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74   * <br><br>
75   * <font color="#004488"><code>
76   * {@code @Override}
77   * <br>
78   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79   * </code></font>
80   *
81   */
82      @Override
83      public finalRefereeFuserRTS_Freeboolean instance() { return new ↩
            finalRefereeFuserRTS_Freeboolean(); }
84  }
```

Source Code: finalRefereeFuserRTS_Openhyperpowerset.java

```
1  /*
2   *   finalRefereeFuserRTS_Openhyperpowerset.java : part of package
3   *         RefereeToolbox; Implementation of fuser based on referee function.
4   ************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeFuserRTS } specialized for
30  * Lattice structures typed {@link finalOpenhyperpowerset }.
31  * An instanciation method {@link finalRefereeFuserRTS_Openhyperpowerset#instance↩
        () } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeFuserRTS_Openhyperpowerset extends RefereeFuserRTS<↩
        finalOpenhyperpowerset,
59                                                                 finalRefereeFuserRTS_Openhyperpowerset↩
            > {
60
61      ////////////////////////////////////////////////:
62      // public part
63      ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
```

```
69   * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
70   * the non generic (typically final) sub-classes</u>.
71   * <br><br>
72   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74   * <br><br>
75   * <font color="#004488"><code>
76   * {@code @Override}
77   * <br>
78   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79   * </code></font>
80   *
81   */
82      @Override
83      public finalRefereeFuserRTS_Openhyperpowerset instance() { return new ↩
            finalRefereeFuserRTS_Openhyperpowerset(); }
84  }
```

Source Code: finalRefereeFuserRTS_Powerset.java

```
1  /*
2   *   finalRefereeFuserRTS_Powerset.java : part of package RefereeToolbox;
3   *          Implementation of fuser based on referee function.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeFuserRTS } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  * An instanciation method {@link finalRefereeFuserRTS_Powerset#instance() } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeFuserRTS_Powerset extends RefereeFuserRTS<finalPowerset,
59                                                                     finalRefereeFuserRTS_Powerset ↩
                                                                        > {
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
69  * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
70  * the non generic (typically final) sub-classes</u>.
```

```
71   * <br><br>
72   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74   * <br><br>
75   * <font color="#004488"><code>
76   * {@code @Override}
77   * <br>
78   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79   * </code></font>
80   *
81   */
82      @Override
83      public finalRefereeFuserRTS_Powerset instance() { return new ↩
            finalRefereeFuserRTS_Powerset(); }
84  }
```

Source Code: finalRefereeFuserRTS_Superpowerset.java

```
1  /*
2   *   finalRefereeFuserRTS_Superpowerset.java : part of package RefereeToolbox;
3   *          Implementation of fuser based on referee function.
4   ****************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeFuserRTS } specialized for
30  * Lattice structures typed {@link finalSuperpowerset }.
31  * An instanciation method {@link finalRefereeFuserRTS_Superpowerset#instance() } ↩
        is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeFuserRTS_Superpowerset extends RefereeFuserRTS<↩
        finalSuperpowerset,
59                                                                 finalRefereeFuserRTS_Superpowerset ↩
                                                                       > {
60
61
62      ////////////////////////////////////////////////:
63      // public part
64      ///////////////
65
66  /**
67   * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
68   * the Class of <i>this</i>).
```

```
69   * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
70   * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
71   * the non generic (typically final) sub-classes</u>.
72   * <br><br>
73   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
74   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
75   * <br><br>
76   * <font color="#004488"><code>
77   * {@code @Override}
78   * <br>
79   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
80   * </code></font>
81   *
82   */
83      @Override
84      public finalRefereeFuserRTS_Superpowerset instance() { return new ↩
            finalRefereeFuserRTS_Superpowerset(); }
85  }
```

Source Code: finalRefereeSampler_Closedhyperpowerset.java

```
1  /*
2   *    finalRefereeSampler_Closedhyperpowerset.java : part of package
3   *          RefereeToolbox; Implement sampled fuser based on referee function.
4   ************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeSampler } specialized for
30  * Lattice structures typed {@link finalClosedhyperpowerset }.
31  * An instanciation method {@link finalRefereeSampler_Closedhyperpowerset#instance↩
        () } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *    RefereeToolbox is free software: you can redistribute it and/or modify
43  *    it under the terms of the GNU General Public License as published by
44  *    the Free Software Foundation, either version 3 of the License, or
45  *    (at your option) any later version.<BR>
46  * <BR>
47  *    RefereeToolbox is distributed in the hope that it will be useful,
48  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *    GNU General Public License for more details.<BR>
51  * <BR>
52  *    You should have received a copy of the GNU General Public License
53  *    along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeSampler_Closedhyperpowerset extends RefereeSampler<↩
        finalClosedhyperpowerset,
59                                                     finalRefereeSampler_Closedhyperpowerset ↩
                                                        >{
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
```

```
 * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
 * the non generic (typically final) sub-classes</u>.
 * <br><br>
 * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
 * definition of {@link BasicBeliefAssignment#instance() } is as follows:
 * <br><br>
 * <font color="#004488"><code>
 * {@code @Override}
 * <br>
 * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
 * </code></font>
 *
 */
   @Override
    public finalRefereeSampler_Closedhyperpowerset instance() { return new ↩
        finalRefereeSampler_Closedhyperpowerset(); }

}
```

Source Code: finalRefereeSampler_Freeboolean.java

```
1  /*
2   *    finalRefereeSampler_Freeboolean.java : part of package RefereeToolbox;
3   *          Implement sampled fuser based on referee function.
4   ****************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeSampler } specialized for
30  * Lattice structures typed {@link finalFreeboolean }.
31  * An instanciation method {@link finalRefereeSampler_Freeboolean#instance() } is
32  * implemented for this non generic class.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *    RefereeToolbox is free software: you can redistribute it and/or modify
43  *    it under the terms of the GNU General Public License as published by
44  *    the Free Software Foundation, either version 3 of the License, or
45  *    (at your option) any later version.<BR>
46  * <BR>
47  *    RefereeToolbox is distributed in the hope that it will be useful,
48  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *    GNU General Public License for more details.<BR>
51  * <BR>
52  *    You should have received a copy of the GNU General Public License
53  *    along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeSampler_Freeboolean extends RefereeSampler<↩
        finalFreeboolean,
59                                                     finalRefereeSampler_Freeboolean>↩
                                                           {
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
69  * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
```

```
70  * the non generic (typically final) sub-classes</u>.
71  * <br><br>
72  * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73  * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74  * <br><br>
75  * <font color="#004488"><code>
76  * {@code @Override}
77  * <br>
78  * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79  * </code></font>
80  *
81  */
82     @Override
83     public finalRefereeSampler_Freeboolean instance() { return new ↩
           finalRefereeSampler_Freeboolean(); }
84  }
```

Source Code: finalRefereeSampler_Openhyperpowerset.java

```
1  /*
2   *   finalRefereeSampler_Openhyperpowerset.java : part of package
3   *         RefereeToolbox; Implement sampled fuser based on referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeSampler } specialized for
30  * Lattice structures typed {@link finalOpenhyperpowerset }.
31  * An instanciation method {@link finalRefereeSampler_Openhyperpowerset#instance()↩
        } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
       Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeSampler_Openhyperpowerset extends RefereeSampler<↩
        finalOpenhyperpowerset,
59                                                     finalRefereeSampler_Openhyperpowerset↩
                                                        >{
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
```

```
69   * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
70   * the non generic (typically final) sub-classes</u>.
71   * <br><br>
72   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74   * <br><br>
75   * <font color="#004488"><code>
76   * {@code @Override}
77   * <br>
78   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79   * </code></font>
80   *
81   */
82     @Override
83      public finalRefereeSampler_Openhyperpowerset instance() { return new ↩
           finalRefereeSampler_Openhyperpowerset(); }
84
85 }
```

Source Code: finalRefereeSampler_Powerset.java

```
1  /*
2   *    finalRefereeSampler_Powerset.java : part of package RefereeToolbox;
3   *          Implement sampled fuser based on referee function.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeSampler } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  * An instanciation method {@link finalRefereeSampler_Powerset#instance() } is
32  * implemented for this non generic class.
33  *
34  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *    RefereeToolbox is free software: you can redistribute it and/or modify
43  *    it under the terms of the GNU General Public License as published by
44  *    the Free Software Foundation, either version 3 of the License, or
45  *    (at your option) any later version.<BR>
46  * <BR>
47  *    RefereeToolbox is distributed in the hope that it will be useful,
48  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *    GNU General Public License for more details.<BR>
51  * <BR>
52  *    You should have received a copy of the GNU General Public License
53  *    along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeSampler_Powerset extends RefereeSampler<finalPowerset,
59                                                   finalRefereeSampler_Powerset>{
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65 /**
66  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
67  * the Class of <i>this</i>).
68  * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
69  * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
70  * the non generic (typically final) sub-classes</u>.
71  * <br><br>
```

```
72   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
73   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
74   * <br><br>
75   * <font color="#004488"><code>
76   * {@code @Override}
77   * <br>
78   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
79   * </code></font>
80   *
81   */
82     @Override
83      public finalRefereeSampler_Powerset instance() { return new ↩
           finalRefereeSampler_Powerset(); }
84
85 }
```

Source Code: finalRefereeSampler_Superpowerset.java

```
1  /*
2   *   finalRefereeSampler_Superpowerset.java : part of package RefereeToolbox;
3   *          Implement sampled fuser based on referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A final instance of {@link RefereeSampler } specialized for
30  * Lattice structures typed {@link finalSuperpowerset }.
31  * An instanciation method {@link finalRefereeSampler_Superpowerset#instance() } ↩
        is
32  * implemented for this non generic class.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class finalRefereeSampler_Superpowerset extends RefereeSampler<↩
        finalSuperpowerset,
59                                                     finalRefereeSampler_Superpowerset ↩
                                                        >{
60
61
62     ////////////////////////////////////////////////:
63     // public part
64     ///////////////
65
66 /**
67  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
68  * the Class of <i>this</i>).
```

```
69   * N.B. {@link BasicBeliefAssignment#instance() } cannot be defined for generic
70   * classes but <u>is necessary</u> for some methods. <u>It has to be defined with
71   * the non generic (typically final) sub-classes</u>.
72   * <br><br>
73   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
74   * definition of {@link BasicBeliefAssignment#instance() } is as follows:
75   * <br><br>
76   * <font color="#004488"><code>
77   * {@code @Override}
78   * <br>
79   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
80   * </code></font>
81   *
82   */
83     @Override
84      public finalRefereeSampler_Superpowerset instance() { return new ↩
           finalRefereeSampler_Superpowerset(); }
85
86 }
```

Source Code: finalSuperpowerset.java

```
1  /*
2   *   finalSuperpowerset.java : part of package RefereeToolbox; Implementation
3   *         of superpowerset.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Final implementation of class {@link Superpowerset }. This class is not generic, and
30  * thus, allows and needs instantiation. The method {@link finalSuperpowerset#instance() }
31  * is implemented.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public final class finalSuperpowerset extends Superpowerset<finalSuperpowerset> {
58
59
60     ////////////////////////////////////////////////:
61     // public part
62     ///////////////
63
64 /**
65  * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
66  * the Class of <i>this</i>).
67  * N.B. {@link Lattice#instance() } cannot be defined
68  * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
69  * defined with the non generic (typically final) sub-classes</u>.
```

```
70   * <br><br>
71   * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
72   * definition of {@link Lattice#instance() } is as follows:
73   * <br><br>
74   * <font color="#004488"><code>
75   * {@code @Override}
76   * <br>
77   * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
78   * </code></font>
79   *
80   */
81      @Override
82      public finalSuperpowerset instance() { return new finalSuperpowerset(); }
83  }
```

Source Code: Freeboolean.java

```
1  /*
2   *    Freeboolean.java : part of package RefereeToolbox; Implementation of Free
3   *          Boolean algebra.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of a Free Boolean structure by means of an array of {@code long↩
        }.
30  * Structural methods are redefined, {@link Freeboolean#size(int)  } and
31  * {@link Freeboolean#size(RefereeToolbox.Freeboolean)  },
32  * and the atomic are defined {@link Freeboolean#atomic(int) }.
33  * The AND and OR operators are inherited from the operators {@code & } and {@code↩
        | }
34  * working on {@code long}.
35  * <i>zero</i> is defined by zeroing all bits. <i>one</i> is defined by setting to
36  * 1 all active bits; the right bits of the {@code long } vector may be inactive,
37  * depending on the size of the Lattice.
38  * The complement/cocomplement operators are inherited from the operators {@code ~↩
        }
39  *  working on {@code long}; exceeding bits are masked by a AND with <i>one</i>.
40  *
41  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
42  *
43  *
44  *
45  * <BR><BR><table border='1' cellPadding='4'><tr><td>
46  * <font color="#008000" style="font-family: georgia">
47  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
48  * <BR>
49  *    RefereeToolbox is free software: you can redistribute it and/or modify
50  *    it under the terms of the GNU General Public License as published by
51  *    the Free Software Foundation, either version 3 of the License, or
52  *    (at your option) any later version.<BR>
53  * <BR>
54  *    RefereeToolbox is distributed in the hope that it will be useful,
55  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
56  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
57  *    GNU General Public License for more details.<BR>
58  * <BR>
59  *    You should have received a copy of the GNU General Public License
60  *    along with RefereeToolbox. If not, see
61  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
62  * </font></table><BR>
63  *
64  */
65 public class Freeboolean<L extends Freeboolean<L>> extends  ArrayBoolean<L>
66                 implements GeneratedLattice<L> {
67
68
```

```
69      protected static int sizeMax=16;
70
71      protected int sizeFrame = -1; // undefined by default
72      protected int sizeSet=-1;
73
74
75      ////////////////////////////////////////////////:
76      // public part
77      ///////////////
78
79  /**
80   * Return the <i>i<SUP>th</SUP></i> atomic proposition.
81   * For class {@link Freeboolean }, the <i>i<SUP>th</SUP></i> atomic proposition
82   * is constituted by all <i>j<SUP>th</SUP></i> bit stored in the
83   * <i>long</i> array such that {@literal j & ( 1 << i ) } is non nul.
84   * <BR><BR>
85   * return <i>null</i> if <i>i</i> is out of bounds.
86   */
87      public L atomic(int i) {
88          if((i>=sizeFrame)||(i<0)) {
89              return null;
90          }
91          int j , index;
92          switch(i) {
93              case 0: for(index=0;index<=size_mem_1;index++)
94                          _memory[index]=0xaaaaaaaaaaaaaaaal; break;
95              case 1:  for(index=0;index<=size_mem_1;index++)
96                          _memory[index]=0xccccccccccccccccl; break;
97              case 2:  for(index=0;index<=size_mem_1;index++)
98                          _memory[index]=0xf0f0f0f0f0f0f0f0l; break;
99              case 3:  for(index=0;index<=size_mem_1;index++)
100                         _memory[index]=0xff00ff00ff00ff00l; break;
101             case 4:  for(index=0;index<=size_mem_1;index++)
102                         _memory[index]=0xffff0000ffff0000l; break;
103             case 5:  for(index=0;index<=size_mem_1;index++)
104                         _memory[index]=0xffffffff00000000l; break;
105             default: j=1<<(i-6);
106                      for(index=0;index<=size_mem_1;index++) {
107                          if((index&j)!=0) _memory[index]=0xffffffffffffffffl;
108                          else _memory[index]=0x0000000000000000l;
109                      }
110                      break;
111         }
112         if(sizeFrame<6) _memory[0]&=highest_long_one;
113         return (L) this;
114     }
115
116 /**
117  * Set or return the size of <i>this</i>.
118  * Diffenrent cases are considered:
119  * <br><br>
120  * <i>newSize<0</i>  --  The size of <i>this</i> is kept unchanged, and this size
121  * is returned by the method.
122  * <br>
123  * <i>newSize>=0</i>  --  If <i>newSize</i> is a valid new size, then the size of
124  * <i>this</i> is changed to <i>newSize</i>, and this size is returned by the ↩
        method.
125  * <br><br>
126  * Valid new size is such that <i>0<=newSize<=sizeMax</i>. By default,
127  * <i>{@link Freeboolean#sizeMax }=16</i>.
128  * <br>
129  * For class {@link Freeboolean }, <i>newSize</i> is stored in protected variable
130  * <i>sizeFrame</i> and is equals to the number of atomic propositions (possibly ↩
        0).
131  * The number of activated bits stored in the <i>long</i> array is equal to <i>2
132  * <sup>newSize</sup>.
133  * <br><br>
134  *
135  * N.B. 1. By default, this method is not implemented by {@link Lattice } directly↩
        ,
136  * but by its subclasses.
137  * <br>
138  * N.B. 2. The size is not necessarilly the cardinal of the lattice, but may be
139  * instead the number of atomic propositions of the lattice.
```

```
140  *
141  */
142      @Override
143      public int size(int newSize) { // if newSize is possible, then change size to ↩
             newSize
144          if((newSize>=0)&&(newSize<=sizeMax)) {
145              sizeFrame=newSize;
146              sizeSet=(1<<sizeFrame);
147              size_mem_1=(sizeSet-1)/64;
148              highest_long_one=sizeSet-64*size_mem_1;
149              highest_long_one=(((1L<<(highest_long_one - 1))-1)<<1)+1;
150              _memory= new long[size_mem_1+1];
151          }
152          return sizeFrame;
153      }
154
155      @Override
156      public L size(L input) {
157          size(input.size(-1));
158          return (L) this;
159      }
160
161  }
```

Source Code: GeneratedLattice.java

```
1  /*
2   *    GeneratedLattice.java : part of package RefereeToolbox; Interface for
3   *          generated lattice.
4   ******************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Contains methods for classes implementing Lattice structures with atomic
30  * elements, {@link #atomic(int) }. Atomics elements are generating elements wich ↩
        are
31  * addressable by means of an index.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *    RefereeToolbox is free software: you can redistribute it and/or modify
42  *    it under the terms of the GNU General Public License as published by
43  *    the Free Software Foundation, either version 3 of the License, or
44  *    (at your option) any later version.<BR>
45  * <BR>
46  *    RefereeToolbox is distributed in the hope that it will be useful,
47  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *    GNU General Public License for more details.<BR>
50  * <BR>
51  *    You should have received a copy of the GNU General Public License
52  *    along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 interface GeneratedLattice<L> extends Lattice<L> {
58
59 /**
60   * Set <i>this</i> to the atomic proposition of index <i>i</i>.
61  * Index <i>i</i> ranges from first value <i>0</i>.
62  * <BR>
63  * N.B. It is necessary to call {@link Lattice#size } or
64  * {@link Lattice#size(java.lang.Object) } before any first use of this method.
65  *
66  */
67     L atomic(int i);
68
69 }
```

Source Code: Lattice.java

```
1  /*
2   *   Lattice.java : part of package RefereeToolbox; Interface for lattice.
3   ******************************************************************************
4   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
5   *
6   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
7   *
8   *   This file is part of RefereeToolbox.
9   *
10  *   RefereeToolbox is free software: you can redistribute it and/or modify
11  *   it under the terms of the GNU General Public License as published by
12  *   the Free Software Foundation, either version 3 of the License, or
13  *   (at your option) any later version.
14  *
15  *   RefereeToolbox is distributed in the hope that it will be useful,
16  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
17  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18  *   GNU General Public License for more details.
19  *
20  *   You should have received a copy of the GNU General Public License
21  *   along with RefereeToolbox.  If not, see <http://www.gnu.org/licenses/>.
22  ******************************************************************************
23  */
24
25 package RefereeToolbox;
26
27 /**
28  * Contains the minimal methods for a class implementing a Lattice structure.
29  *
30  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
31  *
32  *
33  *
34  * <BR><BR><table border='1' cellPadding='4'><tr><td>
35  * <font color="#008000" style="font-family: georgia">
36  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
37  * <BR>
38  *   RefereeToolbox is free software: you can redistribute it and/or modify
39  *   it under the terms of the GNU General Public License as published by
40  *   the Free Software Foundation, either version 3 of the License, or
41  *   (at your option) any later version.<BR>
42  * <BR>
43  *   RefereeToolbox is distributed in the hope that it will be useful,
44  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
45  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
46  *   GNU General Public License for more details.<BR>
47  * <BR>
48  *   You should have received a copy of the GNU General Public License
49  *   along with RefereeToolbox. If not, see
50  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
51  * </font></table><BR>
52  *
53  */
54 public interface Lattice<L> extends Comparable<L>, Cloneable {
55
56 /**
57  * Return a representation of the state of <i>this</i> printed as
58  * a String.
59  *
60  */
61     String state();
62
63 /**
64  * Set or return the size of <i>this</i>.
65  * Diffenrent cases are considered:
66  * <br><br>
67  * <i>newSize<0</i>  --  The size of <i>this</i> is kept unchanged, and this size
68  * is returned by the method.
69  * <br>
70  * <i>newSize>=0</i>  --  If <i>newSize</i> is a valid new size, then the size of
71  * <i>this</i> is changed to <i>newSize</i>, and this size is returned by the ↩
```

```
     method.
72   * <br><br>
73   * N.B. 1. By default, this method is not implemented by {@link Lattice } directly,
74   * but by its subclasses.
75   * <br>
76   * N.B. 2. The size is not necessarilly the cardinal of the lattice, but may be
77   * instead the number of atomic propositions of the lattice.
78   *
79   */
80      int size(int newSize);
81
82  /**
83   * Resize <i>this</i> at the same size than <i>input</i>.
84   *
85   */
86      L size(L input);
87
88   /**
89   * Do an exact copy of input into <i>this</i>.
90   *
91   */
92     L duplicate(L input);
93
94  /**
95   * Create and return an instance of <i>this</i> (i.e. work like a <b>new</b> on
96   * the Class of <i>this</i>). N.B. {@link Lattice#instance() } cannot be defined
97   * for generic classes but <u>is necessary</u> for some methods. <u>It has to be
98   * defined with the non generic (typically final) sub-classes</u>.
99   * <br><br>
100  * For a given non generic sub-class <i>myNonGenericSubclass</i>, a typical
101  * definition of {@link Lattice#instance() } is as follows:
102  * <br><br>
103  * <font color="#004488"><code>
104  * {@code @Override}
105  * <br>
106  * public myNonGenericSubclass instance() { return new myNonGenericSubclass(); }
107  * </code></font>
108  *
109  */
110     L instance();
111
112 /**
113  * Create and return an instance of <i>this</i> and resize it at the same size than <i>this</i>.
114  * This method is actually defined and make a call of {@link Lattice#instance() }.
115  * It does not need to be defined with the non generic sub-classes.
116  *
117  * @see Lattice#instance()
118  *
119  */
120     L instanceNsize();
121
122  /**
123  * Create  a clone of <i>this</i>.
124  *
125  */
126    L clone();
127
128 /**
129  * Set <i>this</i> to zero, the neutral element for
130  * {@link Lattice#or(java.lang.Object, java.lang.Object) }.
131  * N.B. It is necessary to call {@link Lattice#size } or
132  * {@link Lattice#size(java.lang.Object) } before any first use of this
133  * method.
134  *
135  */
136     L zero();
137
138 /**
139  * Set <i>this</i> to one, the neutral element for
140  * {@link Lattice#and(java.lang.Object, java.lang.Object) }.
141  * N.B. It is necessary to call {@link Lattice#size } or
142  * {@link Lattice#size(java.lang.Object) } before any first use of this method.
```

```
143  *
144  */
145     L one();
146
147 /**
148  * Answer <i>true</i> if the proposition stored within <i>this</i> contains the
149  * proposition contained within <i>aProposition</i>. Answer <i>false</i> otherwise↩
         .
150  *
151  */
152     boolean contains(L aProposition);
153
154 /**
155  * Answer <i>true</i> if the proposition stored within <i>this</i> intersects the
156  * proposition contained within <i>aProposition</i> (<i>i.e.</i> the intersection
157  * of both propositions is not <i>zero</i>). Answer <i>false</i> otherwise.
158  *
159  */
160     boolean intersects(L aProposition);
161
162 /**
163  * Compute the AND of <i>left</i> and <i>right</i> and store the result within <i>↩
         this</i>.
164  *
165  */
166     L and(L left, L right);
167
168 /**
169  * Compute the OR of <i>left</i> and <i>right</i> and store the result within <i>↩
         this</i>.
170  *
171  */
172     L or(L left, L right);
173
174 /**
175  * Compare <i>this</i> to <i>aProposition</i> and answer negative integer, zero,
176  * positive integer as <i>this</i> is less than, equal to, or greater than
177  * <i>aProposition</i>.
178  * This comparison is related to a total ordering of the proposition; <em>it is ↩
         not
179  * related to the partial order implied by the logical operators AND and OR.</em>
180  *
181  */
182     int compareTo(L aProposition);
183 }
```

Source Code: LatticeCommon.java

```
1  /*
2   *   LatticeCommon.java : part of package RefereeToolbox; Basical implementation
3   *         of lattice.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Basical implementation of the Latice structure
30  *
31  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
32  *
33  *
34  *
35  * <BR><BR><table border='1' cellPadding='4'><tr><td>
36  * <font color="#008000" style="font-family: georgia">
37  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
38  * <BR>
39  *   RefereeToolbox is free software: you can redistribute it and/or modify
40  *   it under the terms of the GNU General Public License as published by
41  *   the Free Software Foundation, either version 3 of the License, or
42  *   (at your option) any later version.<BR>
43  * <BR>
44  *   RefereeToolbox is distributed in the hope that it will be useful,
45  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
46  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
47  *   GNU General Public License for more details.<BR>
48  * <BR>
49  *   You should have received a copy of the GNU General Public License
50  *   along with RefereeToolbox. If not, see
51  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
52  * </font></table><BR>
53  *
54  */
55 public class LatticeCommon<L extends LatticeCommon<L>>
56         implements Lattice<L>  {
57
58     ////////////////////////////////////////////////:
59     // public part
60     ///////////////
61
62     /**
63      * This method is not defined for class {@link LatticeCommon } and produces
64      * the String {@code "UNDEFINED" }.
65      * <BR><BR>
66      * <b>Documentation inherited from {@link Lattice }:</b><BR>
67      * {@inheritDoc}
68      */
69     public String state() {
70         return "UNDEFINED";
71     }
```

```
72
73      /**
74       * This method is not defined for class {@link LatticeCommon } and generates
75       * an error report and a code exit.
76       * <BR><BR>
77       * <b>Documentation inherited from {@link Lattice }:</b><BR>
78       * {@inheritDoc}
79       */
80      public L zero() {
81          System.err.println("Error :: "+this.getClass().getName() +
82                  ".zero() is not implemented!");
83          System.err.println("Please implement instance() in your class with ↩
                @Override");
84          System.exit(0);
85          return null;
86      }
87
88      /**
89       * This method is not defined for class {@link LatticeCommon } and generates
90       * an error report and a code exit.
91       * <BR><BR>
92       * <b>Documentation inherited from {@link Lattice }:</b><BR>
93       * {@inheritDoc}
94       */
95      public L one() {
96          System.err.println("Error :: "+this.getClass().getName() +
97                  ".one() is not implemented!");
98          System.err.println("Please implement instance() in your class with ↩
                @Override");
99          System.exit(0);
100         return null;
101     }
102
103     /**
104      * This method is not defined for class {@link LatticeCommon } and generates
105      * an error report and a code exit.
106      * <BR><BR>
107      * <b>Documentation inherited from {@link Lattice }:</b><BR>
108      * {@inheritDoc}
109      */
110     public L and(L left, L right) {
111         System.err.println("Error :: "+this.getClass().getName() +
112                 ".and(L,L) is not implemented!");
113         System.err.println("Please implement instance() in your class with ↩
                @Override");
114         System.exit(0);
115         return null;
116     }
117
118     /**
119      * This method is not defined for class {@link LatticeCommon } and generates
120      * an error report and a code exit.
121      * <BR><BR>
122      * <b>Documentation inherited from {@link Lattice }:</b><BR>
123      * {@inheritDoc}
124      */
125     public L or(L left, L right) {
126         System.err.println("Error :: "+this.getClass().getName() +
127                 ".or(L,L) is not implemented!");
128         System.err.println("Please implement instance() in your class with ↩
                @Override");
129         System.exit(0);
130         return null;
131     }
132
133     /**
134      * This method is not defined for class {@link LatticeCommon } and generates
135      * an error report and a code exit.
136      * <BR><BR>
137      * <b>Documentation inherited from {@link Lattice }:</b><BR>
138      * {@inheritDoc}
139      */
140     public  boolean intersects(L aProposition) {
141         System.err.println("Error :: "+this.getClass().getName() +
```

```
142                 ".intersects(L) is not implemented!");
143         System.err.println("Please implement instance() in your class with ↩
                @Override");
144         System.exit(0);
145         return false;
146     }
147 
148     /**
149      * This method is not defined for class {@link LatticeCommon } and generates
150      * an error report and a code exit.
151      * <BR><BR>
152      * <b>Documentation inherited from {@link Lattice }:</b><BR>
153      * {@inheritDoc}
154      */
155     public boolean contains(L aProposition) {
156         System.err.println("Error :: "+this.getClass().getName() +
157                 ".contains(L) is not implemented!");
158         System.err.println("Please implement instance() in your class with ↩
                @Override");
159         System.exit(0);
160         return true;
161     }
162 
163     /**
164      * This method is not defined for class {@link LatticeCommon } and generates
165      * an error report and a code exit.
166      * <BR><BR>
167      * <b>Documentation inherited from {@link Lattice }:</b><BR>
168      * {@inheritDoc}
169      */
170     public int compareTo(L aProposition) {
171         System.err.println("Error :: "+this.getClass().getName() +
172                 ".compareTo(L) is not implemented!");
173         System.err.println("Please implement instance() in your class with ↩
                @Override");
174         System.exit(0);
175         return 0;
176     }
177 
178     /**
179      * This method is not defined for class {@link LatticeCommon } and generates
180      * an error report and a code exit.
181      * <BR><BR>
182      * <b>Documentation inherited from {@link Lattice }:</b><BR>
183      * {@inheritDoc}
184      */
185     public L size(L input) {
186         System.err.println("Error :: "+this.getClass().getName() +
187                 ".size(L) is not implemented!");
188         System.err.println("Please implement instance() in your class with ↩
                @Override");
189         System.exit(0);
190         return null;
191     }
192 
193     /**
194      * This method is not defined for class {@link LatticeCommon } and generates
195      * an error report and a code exit.
196      * <BR><BR>
197      * <b>Documentation inherited from {@link Lattice }:</b><BR>
198      * {@inheritDoc}
199      */
200     public int size(int newSize) {
201         System.err.println("Error :: "+this.getClass().getName() +
202                 ".size(int) is not implemented!");
203         System.err.println("Please implement instance() in your class with ↩
                @Override");
204         System.exit(0);
205         return -1;
206     }
207 
208     /**
209      * This method is not defined for class {@link LatticeCommon } and generates
210      * an error report and a code exit.
```

```
211         * <BR><BR>
212         * <b>Documentation inherited from {@link Lattice }:</b><BR>
213         * {@inheritDoc}
214         */
215        public L duplicate(L input) {
216            System.err.println("Error :: "+this.getClass().getName() +
217                    ".duplicate(L) is not implemented!");
218            System.err.println("Please implement instance() in your class with ↩
                    @Override");
219            System.exit(0);
220            return null;
221        }
222
223        /**
224         * This method is not defined for class {@link LatticeCommon } and generates
225         * an error report and a code exit.
226         * <BR><BR>
227         * <b>Documentation inherited from {@link Lattice }:</b><BR>
228         * {@inheritDoc}
229         */
230        public L instance() {
231            System.err.println("Error :: "+this.getClass().getName() +
232                    ".instance() is not implemented!");
233            System.err.println("Please implement instance() in your class with ↩
                    @Override");
234            System.exit(0);
235            return null;
236        }
237
238        public L instanceNsize() { // create an instance with the same size
239            return instance().size((L) this);
240        }
241
242        @Override
243        public L clone() {
244            return instance().duplicate((L) this);
245        }
246
247    }
```

Source Code: minAssignment.java

```
1  /*
2   *    minAssignment.java : part of package RefereeToolbox; Interface for
3   *          minimal implementation of Basic Belief Assignment.
4   ************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Contains the minimal declaration for classes managing belief assignments.
32  * This interface is implemented by all these classes.
33  *
34  * @see          BasicBeliefAssignment
35  * @see          RefereeFunctionDefault
36  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
37  *
38  *
39  *
40  * <BR><BR><table border='1' cellPadding='4'><tr><td>
41  * <font color="#008000" style="font-family: georgia">
42  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
43  * <BR>
44  *    RefereeToolbox is free software: you can redistribute it and/or modify
45  *    it under the terms of the GNU General Public License as published by
46  *    the Free Software Foundation, either version 3 of the License, or
47  *    (at your option) any later version.<BR>
48  * <BR>
49  *    RefereeToolbox is distributed in the hope that it will be useful,
50  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
51  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
52  *    GNU General Public License for more details.<BR>
53  * <BR>
54  *    You should have received a copy of the GNU General Public License
55  *    along with RefereeToolbox. If not, see
56  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
57  * </font></table><BR>
58  *
59  */
60 public interface minAssignment<Prop extends Lattice<Prop> > {
61
62 /**
63  * Create an array of the assignments stored within the class. These assignments ↩
        not
64  * necessary sum to 1. Notice that the assignments should be positively valued
65  * (focal elements), although <i>this</i> is not a strict requirement.
66  *
67  */
68     ArrayList<Assignment<Prop>> toArray();
69
70 /**
```

```
71   * Get the basic belief assigned to <i>aProposition</i>. May be zero.
72   *
73   */
74      double m(Prop aProposition);
75
76  /**
77   * Get the belief of <i>aProposition</i>. Although not required, it is assumed ↩
       that
78   * the assignments sum to 1.
79   *
80   */
81      double Bel(Prop aProposition);
82
83  /**
84   * Get the plausibility of <i>aProposition</i>. Although not required, it is ↩
       assumed that
85   * the assignments sum to 1.
86   *
87   */
88      double Pl(Prop aProposition);
89
90
91  /**
92   * Search if there is an assignment stored within the class, and which attribute
93   * is equal to the attribute of <i>anAssignment</i>.
94   * Return the found assignment, if there is one. Othewise, return <i>null</i>.
95   *
96   */
97      public Assignment<Prop> findProposition(Assignment<Prop> anAssignment);
98
99  /**
100  * Search if there is an assignment stored within the class, and which attribute
101  * is equal to the proposition <i>aProposition</i>.
102  * Return the found assignment, if there is one. Othewise, return <i>null</i>.
103  *
104  */
105     public Assignment<Prop> findProposition(Prop aProposition);
106
107 }
```

Source Code: Openhyperpowerset.java

```
1  /*
2   *   Openhyperpowerset.java : part of package RefereeToolbox; Implementation
3   *         of open hyperpowerset.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of an open hyperpowerset structure. An open hyperpowerset is an
30  * hyperpowerset defined on the basis of a free boolean algebra: it is generated
31  * from the atomic propositions of the free boolean algebra but without the use of
32  * the complement operator of the free boolean algebra.
33  * <BR><BR>
34  * The open hyperpowerset implies an open world hypothesis, that is:
35  *<BR><BR>
36  * <i> <b>OR</b><sub>0 =< i < sizeFrame</sub> atomic(i)  !=  one</i>
37  * <BR><BR>
38  * Although generated without the complement of the free Boolean algebra, the open
39  * hyperpowerset actually has proper complement and cocomplement operators.
40  * The complement and cocomplement are distinct operators for the open ↩
        hyperpowerset.
41  *
42  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
43  *
44  *
45  *
46  * <BR><BR><table border='1' cellPadding='4'><tr><td>
47  * <font color="#008000" style="font-family: georgia">
48  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
49  * <BR>
50  *   RefereeToolbox is free software: you can redistribute it and/or modify
51  *   it under the terms of the GNU General Public License as published by
52  *   the Free Software Foundation, either version 3 of the License, or
53  *   (at your option) any later version.<BR>
54  * <BR>
55  *   RefereeToolbox is distributed in the hope that it will be useful,
56  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
57  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
58  *   GNU General Public License for more details.<BR>
59  * <BR>
60  *   You should have received a copy of the GNU General Public License
61  *   along with RefereeToolbox. If not, see
62  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
63  * </font></table><BR>
64  *
65  */
66 public class Openhyperpowerset<L extends Openhyperpowerset<L>> extends Freeboolean↩
        <L> {
67     protected L theZero = null;
68     protected L theOne = null;
69
```

```
70      /**
71       * Compute the <b>complement</b> of <i>aProposition</i> and store the result
72       * within <i>this</i>.
73       * The complement operator for the open hyperpowerset is defined by:
74       * <BR><BR>
75       * <i> complement(zero) = one</i>
76       * <BR>
77       * <i> complement(X) = zero</i> if <i>X != zero</i>
78       *
79       */
80      @Override
81      public L complement(L aProposition) {
82          if(theZero==null) {
83              theZero = instanceNsize().zero();
84              theOne = instanceNsize().one();
85          }
86          if(aProposition.compareTo(theZero)==0) one();
87          else zero();
88          return (L) this;
89      }
90
91      /**
92       * Compute the <b>complement</b> of <i>this</i> and store the result within
93       * <i>this</i>.
94       * The complement operator for the open hyperpowerset is defined by:
95       * <BR><BR>
96       * <i> complement(zero) = one</i>
97       * <BR>
98       * <i> complement(X) = zero</i> if <i>X != zero</i>
99       *
100      */
101     @Override
102     public L complement() {
103         if(theZero==null) {
104             theZero = instanceNsize().zero();
105             theOne = instanceNsize().one();
106         }
107         if(compareTo(theZero)==0) one();
108         else zero();
109         return (L) this;
110     }
111
112     /**
113      * Compute the <b>cocomplement</b> of <i>aProposition</i> and store the result
114      * within <i>this</i>.
115      * The cocomplement operator for the open hyperpowerset is defined by:
116      * <BR><BR>
117      * <i> cocomplement(one) = zero</i>
118      * <BR>
119      * <i> cocomplement(X) = one</i> if <i>X != one</i>
120      *
121      */
122     @Override
123     public L cocomplement(L aProposition) {
124         if(theZero==null) {
125             theZero = instanceNsize().zero();
126             theOne = instanceNsize().one();
127         }
128         if(aProposition.compareTo(theOne)==0) zero();
129         else one();
130         return (L) this;
131     }
132
133     /**
134      * Compute the <b>cocomplement</b> of <i>this</i> and store the result
135      * within <i>this</i>.
136      * The cocomplement operator for the open hyperpowerset is defined by:
137      * <BR><BR>
138      * <i> cocomplement(one) = zero</i>
139      * <BR>
140      * <i> cocomplement(X) = one</i> if <i>X != one</i>
141      *
142      */
143     @Override
```

```
144     public L cocomplement() {
145         if(theZero==null) {
146             theZero = instanceNsize().zero();
147             theOne = instanceNsize().one();
148         }
149         if(compareTo(theOne)==0) zero();
150         else one();
151         return (L) this;
152     }
153 
154 }
```

Source Code: Powerset.java

```
1  /*
2   *   Powerset.java : part of package RefereeToolbox; Implementation of powerset.
3   ******************************************************************************
4   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
5   *
6   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
7   *
8   *   This file is part of RefereeToolbox.
9   *
10  *   RefereeToolbox is free software: you can redistribute it and/or modify
11  *   it under the terms of the GNU General Public License as published by
12  *   the Free Software Foundation, either version 3 of the License, or
13  *   (at your option) any later version.
14  *
15  *   RefereeToolbox is distributed in the hope that it will be useful,
16  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
17  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18  *   GNU General Public License for more details.
19  *
20  *   You should have received a copy of the GNU General Public License
21  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
22  ******************************************************************************
23  */
24
25 package RefereeToolbox;
26
27 /**
28  * Implementation of a Powerset structure by means of an array of {@code long}.
29  * Structural methods are redefined, {@link Freeboolean#size(int)  } and
30  * {@link Freeboolean#size(RefereeToolbox.Freeboolean)  },
31  * and the atomic are defined {@link Freeboolean#atomic(int) }.
32  * The AND and OR operators are inherited from the operators {@code & } and {@code↩
        | }
33  * working on {@code long}.
34  * <i>zero</i> is defined by zeroing all bits. <i>one</i> is defined by setting to
35  * 1 all active bits; the right bits of the {@code long } vector may be inactive,
36  * depending on the size of the Lattice.
37  * The complement/cocomplement operators are inherited from the operators {@code ~↩
        }
38  *  working on {@code long}; exceeding bits are masked by a AND with <i>one</i>.
39  *
40  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
       Dambreville</A>
41  *
42  *
43  *
44  * <BR><BR><table border='1' cellPadding='4'><tr><td>
45  * <font color="#008000" style="font-family: georgia">
46  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
47  * <BR>
48  *   RefereeToolbox is free software: you can redistribute it and/or modify
49  *   it under the terms of the GNU General Public License as published by
50  *   the Free Software Foundation, either version 3 of the License, or
51  *   (at your option) any later version.<BR>
52  * <BR>
53  *   RefereeToolbox is distributed in the hope that it will be useful,
54  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
55  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
56  *   GNU General Public License for more details.<BR>
57  * <BR>
58  *   You should have received a copy of the GNU General Public License
59  *   along with RefereeToolbox. If not, see
60  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
61  * </font></table><BR>
62  *
63  */
64 public class Powerset<L extends Powerset<L>> extends ArrayBoolean<L>
65         implements GeneratedLattice<L> {
66
67     protected static int sizeMax=1<<16;
68
69     protected int sizeFrame = -1; // Frame of discernmment is undefined by default
```

```
70
71
72      ////////////////////////////////////////////////:
73      // public part
74      ///////////////
75
76  /**
77   * Return the <i>i<SUP>th</SUP></i> atomic proposition.
78   * For class {@link Powerset }, the <i>i<SUP>th</SUP></i> atomic proposition is
79   * constituted by the <i>i<SUP>th</SUP></i> bit stored in the
80   * <i>long</i> array.
81   * <BR><BR>
82   * return <i>null</i> if <i>i</i> is out of bounds.
83   */
84      public L atomic(int i) {
85          if((i>=sizeFrame)||(i<0)) {
86              return null;
87          }
88          zero();
89          int offset = i%64;
90          int index = i/64;
91          _memory[index]=( 1l << offset);
92          return (L) this;
93      }
94
95  /**
96   * Set or return the size of <i>this</i>.
97   * Diffenrent cases are considered:
98   * <br><br>
99   * <i>newSize<0</i>  --  The size of <i>this</i> is kept unchanged, and this size
100  * is returned by the method.
101  * <br>
102  * <i>newSize>=0</i>  --  If <i>newSize</i> is a valid new size, then the size of
103  * <i>this</i> is changed to <i>newSize</i>, and this size is returned by the ↩
       method.
104  * <br><br>
105  * Valid new size is such that <i>1<=newSize<=sizeMax</i>. By default,
106  * <i>{@link Powerset#sizeMax }=1<<16</i>.
107  * <br>
108  * For class {@link Powerset }, <i>newSize</i> is stored in protected variable
109  * <i>sizeFrame</i> and is equals to the number <em>(has to be >1)</em> of ↩
       activated
110  * bits stored in the <i>long</i> array.
111  * Moreover, each activated bit constitute an atomic proposition.
112  * <br><br>
113  *
114  * N.B. 1. By default, this method is not implemented by {@link Lattice } directly↩
       ,
115  * but by its subclasses.
116  * <br>
117  * N.B. 2. The size is not necessarilly the cardinal of the lattice, but may be
118  * instead the number of atomic propositions of the lattice.
119  *
120  */
121     @Override
122     public int size(int newSize) { // if newSize is possible, then change size to ↩
           newSize
123         if((newSize>0)&&(newSize<=sizeMax)) {
124             sizeFrame=newSize;
125             size_mem_1=(sizeFrame-1)/64;
126             highest_long_one=sizeFrame-64*size_mem_1;
127             highest_long_one=(((1L<<(highest_long_one - 1))-1)<<1)+1;
128             //
129             _memory= new long[size_mem_1+1];
130         }
131         return sizeFrame;
132     }
133
134     @Override
135     public L size(L input) {
136         size(input.size(-1));
137         return (L) this;
138     }
139
```

```
140 }
```

Source Code: RefereeFunctionDefault.java

```
1  /*
2   *    RefereeFunctionDefault.java : part of package RefereeToolbox; Default
3   *          implementation of referee function.
4   ********************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * This class implements the default Referee Function.
32  * This default Referee Function actually encodes the combination rule of Dempster↩
        -Shafer.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *    RefereeToolbox is free software: you can redistribute it and/or modify
43  *    it under the terms of the GNU General Public License as published by
44  *    the Free Software Foundation, either version 3 of the License, or
45  *    (at your option) any later version.<BR>
46  * <BR>
47  *    RefereeToolbox is distributed in the hope that it will be useful,
48  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *    GNU General Public License for more details.<BR>
51  * <BR>
52  *    You should have received a copy of the GNU General Public License
53  *    along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public class RefereeFunctionDefault<Prop extends Lattice<Prop> > {
59
60
61      ////////////////////////////////////////////////:
62      // public part
63      ///////////////
64
65 /**
66  * Implement the Referee Function of the class.
67  * The referee function takes as context:
68  * <BR><BR>
69  * (a) The list of the propositions proposed by each source of information,
70  * <BR>
```

```
71  * (b) The list of basic belief assigned by each sources to their respective proposal,
72  * <BR>
73  * (c) The list of Basic Belief Assignmment of the sources. This last information,
74  * which constitutes a more general and global information of context, is rarely
75  * used in practice, but is provided for the sack of the generality of the implementation.
76  * <BR><BR>
77  * Information (a) and (b) are provided by variable <i>assignIn</i>.
78  * Information (c) are provided by variable <i>bbaIn</i>.
79  * <BR><BR>
80  * The referee function returns a list of assignments which sum to 1 and models a
81  * probabilistic final decision.
82  *
83  * @param assignIn
84  *              The list of propositions proposed by the sources together with their
85  * @param bbaIn
86  *              The list of Basic Belief Assignmment of the sources
87  * assigned basic belief
88  * @return A list of assignments modelling a probabilistic decision
89  *
90  */
91     public ArrayList<Assignment<Prop>> refereeFunction(ArrayList<Assignment<Prop>>
           assignIn,
92                                                        ArrayList<minAssignment<
                                                               Prop>> bbaIn) {
93         if(assignIn==null) {
94             System.err.println("Error :: "+this.getClass().getName() +
95                 ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
96             System.err.println("Error :: assignIn is null");
97             System.exit(0);
98         }
99         if(assignIn.size()<1) {
100            System.err.println("Error :: "+this.getClass().getName() +
101                ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
102            System.err.println("Error :: assignIn is empty");
103            System.exit(0);
104        }
105
106        int i;
107        ArrayList<Assignment<Prop>> arbitrament=new ArrayList<Assignment<Prop>>();
108        Assignment<Prop> finalAssignment = new Assignment<Prop>();
109        Prop fusedProp=assignIn.get(0).attribute.clone();
110        finalAssignment.attribute=fusedProp;
111        finalAssignment.value=1.;
112        arbitrament.add(finalAssignment);
113
114        for(i=1;i<assignIn.size();i++) {
115            fusedProp.and(fusedProp,assignIn.get(i).attribute);
116        }
117        return arbitrament;
118    }
119
120 }
```

Source Code: RefereeFunctionDempster.java

```
1  /*
2   *    RefereeFunctionDempster.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ********************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * This class implements the Referee Function encoding the combination rule of
30  * Dempster-Shafer.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *    RefereeToolbox is free software: you can redistribute it and/or modify
41  *    it under the terms of the GNU General Public License as published by
42  *    the Free Software Foundation, either version 3 of the License, or
43  *    (at your option) any later version.<BR>
44  * <BR>
45  *    RefereeToolbox is distributed in the hope that it will be useful,
46  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *    GNU General Public License for more details.<BR>
49  * <BR>
50  *    You should have received a copy of the GNU General Public License
51  *    along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RefereeFunctionDempster<Prop extends Lattice<Prop> >
57                         extends RefereeFunctionDefault<Prop> {
58                         // Nothing to do; Dempster is by default!
59 }
```

Source Code: RefereeFunctionDisjunctive.java

```
1  /*
2   *    RefereeFunctionDisjunctive.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ********************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29 /**
30  * This class implements the Referee Function encoding the disjunctive combination
31  * rule.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *    RefereeToolbox is free software: you can redistribute it and/or modify
42  *    it under the terms of the GNU General Public License as published by
43  *    the Free Software Foundation, either version 3 of the License, or
44  *    (at your option) any later version.<BR>
45  * <BR>
46  *    RefereeToolbox is distributed in the hope that it will be useful,
47  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *    GNU General Public License for more details.<BR>
50  * <BR>
51  *    You should have received a copy of the GNU General Public License
52  *    along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public class RefereeFunctionDisjunctive<Prop extends Lattice<Prop> >
58                         extends RefereeFunctionDefault<Prop> {
59
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65     @Override
66     public ArrayList<Assignment<Prop>> refereeFunction(ArrayList<Assignment<Prop>>↩
           assignIn,
67                                                        ArrayList<minAssignment<Prop>>↩
                                                             bbaIn) {
68         if(assignIn==null) {
69             System.err.println("Error :: "+this.getClass().getName() +
```

```
70                  ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
71              System.err.println("Error :: assignIn is null");
72              System.exit(0);
73          }
74          if(assignIn.size()<1) {
75              System.err.println("Error :: "+this.getClass().getName() +
76                  ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
77              System.err.println("Error :: assignIn is empty");
78              System.exit(0);
79          }
80
81          int i;
82          ArrayList<Assignment<Prop>> arbitrament=new ArrayList<Assignment<Prop>>();
83          Assignment<Prop> finalAssignment = new Assignment<Prop>();
84          Prop fusedProp=assignIn.get(0).attribute.clone();
85          finalAssignment.attribute=fusedProp;
86          finalAssignment.value=1.;
87          arbitrament.add(finalAssignment);
88
89          for(i=1;i<assignIn.size();i++) {
90              fusedProp.or(fusedProp,assignIn.get(i).attribute);
91          }
92          return arbitrament;
93      }
94
95  }
```

Source Code: RefereeFunctionDuboisPrade.java

```
1  /*
2   *   RefereeFunctionDuboisPrade.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29 /**
30  * This class implements the Referee Function encoding the combination rule of ↩
        Dubois
31  * & Prade.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public class RefereeFunctionDuboisPrade<Prop extends Lattice<Prop> >
58                                     extends RefereeFunctionDefault<Prop> {
59
60
61     ////////////////////////////////////////////////:
62     // public part
63     ///////////////
64
65     @Override
66     public ArrayList<Assignment<Prop>> refereeFunction(ArrayList<Assignment<Prop>>↩
           assignIn,
67                                                        ArrayList<minAssignment<Prop>>↩
                                                              bbaIn) {
68         if(assignIn==null) {
```

```
69              System.err.println("Error :: "+this.getClass().getName() +
70                  ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
71              System.err.println("Error :: assignIn is null");
72              System.exit(0);
73          }
74          if(assignIn.size()<1) {
75              System.err.println("Error :: "+this.getClass().getName() +
76                  ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
77              System.err.println("Error :: assignIn is empty");
78              System.exit(0);
79          }
80
81          int i;
82          ArrayList<Assignment<Prop>> arbitrament=new ArrayList<Assignment<Prop>>();
83          Assignment<Prop> finalAssignment = new Assignment<Prop>();
84          Prop fusedPropConj=assignIn.get(0).attribute.clone();
85          Prop fusedPropDisj=fusedPropConj.instanceNsize().zero();
86          finalAssignment.value=1.;
87          for(i=1;i<assignIn.size();i++) fusedPropConj.and(fusedPropConj,assignIn.↩
                get(i).attribute);
88          if(fusedPropConj.compareTo(fusedPropDisj)==0) {
89              for(i=0;i<assignIn.size();i++) fusedPropDisj.or(fusedPropDisj,assignIn↩
                    .get(i).attribute);
90              finalAssignment.attribute=fusedPropDisj;
91          } else finalAssignment.attribute=fusedPropConj;
92          arbitrament.add(finalAssignment);
93          return arbitrament;
94      }
95
96  }
```

Source Code: RefereeFunctionPCR6.java

```
1  /*
2   *   RefereeFunctionPCR6.java : part of package RefereeToolbox; Implementation
3   *         of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29 /**
30  * This class implements the Referee Function encoding the PCR6 combination rule.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RefereeFunctionPCR6<Prop extends Lattice<Prop> >
57                                     extends RefereeFunctionDefault<Prop> {
58
59
60     ////////////////////////////////////////////////:
61     // public part
62     ///////////////
63
64     @Override
65     public ArrayList<Assignment<Prop>> refereeFunction(ArrayList<Assignment<Prop>>↩
           assignIn,
66                                                        ArrayList<minAssignment<↩
                                                             Prop>> bbaIn) {
67         if(assignIn==null) {
68             System.err.println("Error :: "+this.getClass().getName() +
69                 ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
```

```
70              System.err.println("Error :: assignIn is null");
71              System.exit(0);
72          }
73          if(assignIn.size()<1) {
74              System.err.println("Error :: "+this.getClass().getName() +
75                  ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
76              System.err.println("Error :: assignIn is empty");
77              System.exit(0);
78          }
79
80          int i;
81          double Norm;
82          ArrayList<Assignment<Prop>> arbitrament=new ArrayList<Assignment<Prop>>();
83          Assignment<Prop> finalAssignment;
84          Prop fusedProp=assignIn.get(0).attribute.clone();
85          Prop zero=fusedProp.instanceNsize().zero();
86          Norm=0.;
87          for(i=1;i<assignIn.size();i++) {
88              fusedProp.and(fusedProp,assignIn.get(i).attribute);
89          }
90          if(fusedProp.compareTo(zero)==0) {
91              for(i=0;i<assignIn.size();i++) {
92                  Norm+=assignIn.get(i).value;
93              }
94              for(i=0;i<assignIn.size();i++) {
95                  finalAssignment = new Assignment<Prop>();
96                  finalAssignment.value=assignIn.get(i).value/Norm;
97                  finalAssignment.attribute=assignIn.get(i).attribute.clone();
98                  arbitrament.add(finalAssignment);
99              }
100         } else {
101             finalAssignment = new Assignment<Prop>();
102             finalAssignment.value=1.;
103             finalAssignment.attribute=fusedProp;
104             arbitrament.add(finalAssignment);
105         }
106         return arbitrament;
107     }
108
109 }
```

Source Code: RefereeFunctionPCRSharp.java

```
1  /*
2   *   RefereeFunctionPCRSharp.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * This class implements the Referee Function encoding the PCRSharp combination ↩
      rule.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *   RefereeToolbox is free software: you can redistribute it and/or modify
42  *   it under the terms of the GNU General Public License as published by
43  *   the Free Software Foundation, either version 3 of the License, or
44  *   (at your option) any later version.<BR>
45  * <BR>
46  *   RefereeToolbox is distributed in the hope that it will be useful,
47  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *   GNU General Public License for more details.<BR>
50  * <BR>
51  *   You should have received a copy of the GNU General Public License
52  *   along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public class RefereeFunctionPCRSharp<Prop extends Lattice<Prop> >
58                                 extends RefereeFunctionDefault<Prop> {
59     protected int theEntrySize=-1;
60     protected Prop theDefault=null;
61     protected Prop theZero=null;
62     protected Prop theOne=null;
63     protected  boolean[] theMask=null;
64     protected  boolean[] theRevMask=null;
65
66     protected  double[] weightTab=null;
67     protected  int[] mapTab=null;
68     protected ArrayList<Prop> thePropositionTab=null;
69
70     protected  ArrayList<Assignment<Prop>> getWeightedSubset(int subsetSize,
```

```
71                                          ArrayList<Assignment<Prop>> assignIn) {
72          double theNorm;
73          int j,k,pivot;
74          boolean justMoved;
75          ArrayList<Assignment<Prop>> theAssignmentList;
76          theAssignmentList = new ArrayList<Assignment<Prop>>();
77          Assignment<Prop> curAssignment;
78          Prop aProposition;
79          double aValue;
80
81          // init
82          theNorm=0.;
83          pivot=subsetSize;
84          weightTab[pivot]=1.;
85          thePropositionTab.set(pivot, theOne);
86          mapTab[pivot]=theEntrySize;
87          justMoved=true;
88          // init done
89
90          // loop
91          do {
92              if(justMoved) {
93                  justMoved=false;
94                  for(j=pivot;j>0;) {
95                      k=j;
96                      j--;
97                      mapTab[j]=j;
98                      weightTab[j]=weightTab[k]*assignIn.get(j).value;
99                      thePropositionTab.set(j, theZero.instanceNsize().and(
100                         assignIn.get(j).attribute, thePropositionTab.get(k)));
101                     pivot--;
102                 }
103                 // build a new assignment?
104                 aProposition=thePropositionTab.get(0);
105                 if(aProposition.compareTo(theZero)!=0) {
106                     // a new assignment is created for the next case
107                     aValue=weightTab[0];
108                     curAssignment = new Assignment<Prop>();
109                     curAssignment.attribute=aProposition;
110                     curAssignment.value=aValue;
111                     theAssignmentList.add(curAssignment);
112                     theNorm+=aValue;
113                 }
114             }
115             k=pivot+1;
116             j=mapTab[pivot]+1;
117             if(j<mapTab[k]) {
118                 mapTab[pivot]=j;
119                 weightTab[pivot]=weightTab[k]*assignIn.get(j).value;
120                 thePropositionTab.set(pivot, theZero.instanceNsize().and(
121                         assignIn.get(j).attribute, thePropositionTab.get(k)));
122                 justMoved=true;
123             } else pivot++;
124
125         } while(pivot<subsetSize);
126
127         for(j=0;j<theAssignmentList.size();j++) theAssignmentList.get(j).value/=↩
                theNorm;
128
129         return theAssignmentList;
130     }
131
132
133     protected ArrayList<Assignment<Prop>> askWeightedSubset(int subsetSize,
134                                             ArrayList<Assignment<Prop>> assignIn) {
135                 // Test the mask in order to decide for the computation
136         int k;
137         if(theMask==null) {
138             if(theRevMask==null) return getWeightedSubset(subsetSize,assignIn);
139         } else {
140             if((k=(subsetSize-1))<theMask.length)
141                     if(theMask[k]) return getWeightedSubset(subsetSize,assignIn);
142         }
143         if(theRevMask!=null) {
```

```
144             if((k=(theEntrySize-subsetSize))<theRevMask.length)
145                     if(theRevMask[k]) return getWeightedSubset(subsetSize,assignIn↩
                            );
146         }
147         return new ArrayList<Assignment<Prop>>();
148     }
149 
150 
151     ////////////////////////////////////////////:
152     // public part
153     ///////////////
154 
155 /**
156  * Set the default answer to proposition <i>aDefault</i>.
157  * The default answer is returned by the PCRSharp referee function, when no ↩
        consensus
158  * have been found among the acceptable subset combination of sources.
159  * <BR><BR>
160  * The default answer is initialized to <i>null</i> at the class construction,
161  * which is automatically handled as a <i>zero</i>.
162  *
163  */
164     public void setDefaultAnswer(Prop aDefault) {
165         if(aDefault==null) {
166             System.err.println("Error :: "+this.getClass().getName() +
167                 ".setDefaultAnswer(Prop)");
168              System.err.println("Error :: aDefault is null");
169              System.exit(0);
170         }
171         theDefault=aDefault.clone();
172     }
173 
174 /**
175  * Set a mask and a reverse mask which will select the sizes of subset combination
176  * (of the sources entries) which are taken into consideration.
177  * These parameters work as follows:
178  * <br><br>
179  * If <i>aMask=null</i> and <i>aRevMask=null</i>\,, then all subset combinations ↩
        are
180  * managed (this is a dangerous choice, when there are many sources).
181  * <br><br>Otherwise:
182  * <br>
183  * If <i>aMask=null</i>, then <i>aMask</i> is not considered for masking.
184  * <br>
185  * If <i>aRevMask=null</i>, then <i>aRevMask</i> is not considered for masking.
186  * <br><br>
187  * aMask defined the clearance where it is <i>true</i> from (minimal) subset size
188  * 1 upward
189  * <br>
190  * aRevMask defined the clearance where it is <i>true</i> from (maximal) subset
191  * size <i>theEntrySize</i> downward
192  *
193  */
194     public void setMask(boolean[] aMask, boolean[] aRevMask) {
195         // NB: null Mask means that the mask is not considered
196         // null & null masks means that there is no masking
197         // aMask => define clearance from (minimal) set size 1 upward
198         // aRevMask => define clearance from (maximal) set size theEntrySize ↩
                downward
199         int i;
200         theMask=new boolean[aMask.length];
201         theRevMask=new boolean[aRevMask.length];
202         for(i=0;i<theMask.length;i++) theMask[i]=aMask[i];
203         for(i=0;i<theRevMask.length;i++) theRevMask[i]=aRevMask[i];
204     }
205 
206     @Override
207     public ArrayList<Assignment<Prop>> refereeFunction(ArrayList<Assignment<Prop>>↩
            assignIn,
208                                                        ArrayList<minAssignment<Prop↩
                                                               >> bbaIn) {
209         if(assignIn==null) {
210             System.err.println("Error :: "+this.getClass().getName() +
211                 ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
```

```
212                 System.err.println("Error :: assignIn is null");
213                 System.exit(0);
214             }
215             int i;
216             i=assignIn.size();
217             if(i<1) {
218                 System.err.println("Error :: "+this.getClass().getName() +
219                     ".refereeFunction(ArrayList<Assignment<Prop>>,ArrayList<B>)");
220                 System.err.println("Error :: assignIn is empty");
221                 System.exit(0);
222             }
223             if(theEntrySize!=i) {
224                 theEntrySize=i;
225                 weightTab = new double[theEntrySize+1];
226                 mapTab= new int[theEntrySize+1];
227                 thePropositionTab = new ArrayList<Prop>();
228                 for(i=0;i<=theEntrySize;i++) {
229                     thePropositionTab.add(null);
230                 }
231             }
232             if(theZero==null) {
233                 theZero=assignIn.get(0).attribute.instanceNsize().zero();
234                 theOne=theZero.instanceNsize().one();
235             }
236             if(theDefault==null) theDefault=theZero;
237             ArrayList<Assignment<Prop>> arbitrament=null;
238             for(i=theEntrySize;i>0;i--) {
239                 arbitrament=askWeightedSubset(i, assignIn);
240                 if(arbitrament.size()>0) return arbitrament;
241             }
242             Assignment<Prop> finalAssignment = new Assignment<Prop>();
243             finalAssignment.attribute=theDefault.clone();
244             finalAssignment.value=1.;
245             // NB: initialization of arbitrament is done in the loop
246             arbitrament.add(finalAssignment);
247             return arbitrament;
248         }
249
250 }
```

Source Code: RefereeFuserRTS.java

```
1  /*
2   *   RefereeFuserRTS.java : part of package RefereeToolbox; Implementation of
3   *          fuser by means of referee function.
4   ********************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ********************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Relaxed extension of {@link TreeSetRefereeFuser }.
30  *
31  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
32  *
33  *
34  *
35  * <BR><BR><table border='1' cellPadding='4'><tr><td>
36  * <font color="#008000" style="font-family: georgia">
37  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
38  * <BR>
39  *   RefereeToolbox is free software: you can redistribute it and/or modify
40  *   it under the terms of the GNU General Public License as published by
41  *   the Free Software Foundation, either version 3 of the License, or
42  *   (at your option) any later version.<BR>
43  * <BR>
44  *   RefereeToolbox is distributed in the hope that it will be useful,
45  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
46  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
47  *   GNU General Public License for more details.<BR>
48  * <BR>
49  *   You should have received a copy of the GNU General Public License
50  *   along with RefereeToolbox. If not, see
51  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
52  * </font></table><BR>
53  *
54  */
55 public class RefereeFuserRTS < Prop extends Lattice<Prop>,
56                                B extends RefereeFuserRTS<Prop,B>>
57                                    extends TreeSetRefereeFuser<Prop,B>
58                                          implements RelaxedBBA<Prop,B> {
59
60     @Override
61     protected boolean relaxed() { return true; }
62
63
64     ////////////////////////////////////////////////:
65     // public part
66     ///////////////
67
68 /**
69  * Relax <i>this</i> so that it contains no more than <i>maxMem</i> assignments.
70  *
71  */
```

```
72      @Override
73      public boolean relax(int maxMem) { return super.relax(maxMem); }
74
75  /**
76   * Relax <i>this</i> so that it contains no more than {@link RelaxedBBA#maxSize(↩
        int) }
77   * assignments.
78   *
79   */
80      @Override
81      public boolean relax() { return super.relax(); }
82
83  /**
84   * Set or return the default maximum number of assignments stored in <i>this</i>,
85   * depending on parameter <i>newSize</i>.
86   * If <i>newSize>0</i>, then set the default maximum number of assignments to <i>↩
        newSize</i>.
87   * In any case, return the actual value of the default maximum number.
88   * <BR><BR>
89   * N.B. It is not possible to exceed this number of assignments, except for ↩
        temporary
90   * computed assignments.
91   *
92   */
93      @Override
94      public int maxSize(int newSize) { return super.maxSize(newSize); }
95  }
```

Source Code: RefereeSampler.java

```
1  /*
2   *    RefereeSampler.java : part of package RefereeToolbox; Implementation of
3   *          sampling-based fuser by means of referee function.
4   *****************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  *****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * This class implements a generic combination of Basic Belief Assignments by the
32  * means of referee functions and on the basis of a particle approximation. This
33  * class does not handle direct rule implementations, {@link BBAFuser}, or exact
34  * referee-based fusion {@link BBARefereeFuser}.
35  *
36  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
37  *
38  *
39  *
40  * <BR><BR><table border='1' cellPadding='4'><tr><td>
41  * <font color="#008000" style="font-family: georgia">
42  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
43  * <BR>
44  *    RefereeToolbox is free software: you can redistribute it and/or modify
45  *    it under the terms of the GNU General Public License as published by
46  *    the Free Software Foundation, either version 3 of the License, or
47  *    (at your option) any later version.<BR>
48  * <BR>
49  *    RefereeToolbox is distributed in the hope that it will be useful,
50  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
51  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
52  *    GNU General Public License for more details.<BR>
53  * <BR>
54  *    You should have received a copy of the GNU General Public License
55  *    along with RefereeToolbox. If not, see
56  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
57  * </font></table><BR>
58  *
59  */
60 public class RefereeSampler<Prop extends Lattice<Prop>,
61                                         sB extends RefereeSampler<Prop,sB> >
62                                             extends TreeSetBBA<Prop,sB>
63                                             implements SampledBBARefereeFuser<Prop↩
                                                  ,sB> {
64
65     protected Prop theZero=null;
66
67     protected RefereeFunctionDefault<Prop> theRefereeFunction;
68
69     protected static Random rng = new Random();
70
```

```
71      protected boolean __updated=true;
72
73      protected double[] cumulativeDistribution_Sampler=null; // cumulative ↩
            distribbution is used
74                                              // for sampling
75      protected double[] cumulativeDistribution_Mixer=null; // cumulative ↩
            distribbution is used
76                                              // for sampling
77
78      protected ArrayList<Assignment<Prop>> mappedAttribute;
79
80      protected ArrayList<Assignment<Prop>> answeredAttribute;
81
82      protected int[] jumpTable_Sampler=null; // used for ensuring fast sampling
83
84      protected int[] jumpTable_Mixer=null; // used for ensuring fast sampling
85
86      protected ArrayList<minAssignment<Prop>> BBASources= new ArrayList<↩
            minAssignment<Prop>>();
87
88      protected ArrayList<sB> fuserSources;
89
90      protected ArrayList<sB> mixerSources;
91
92
93      protected boolean setTables_sampler() {
94          Iterator iter = bbaTreeVA.iterator();
95          int size=bbaTreeVA.size();
96          cumulativeDistribution_Sampler=new double[size];
97          jumpTable_Sampler=new int[size];
98          mappedAttribute= new ArrayList<Assignment<Prop>>();
99          Assignment<Prop> assignment;
100         int i; // true variable
101         int k; // uniform variable
102         double Cumul=0.; // For computing law cumulatives
103         double uStep=1./size; // uniform step
104
105         for(i=0; iter.hasNext(); i++) { // compute the cumulative of inDist
106                 assignment=(Assignment<Prop>) iter.next();
107                 mappedAttribute.add(assignment);
108                 Cumul+=assignment.value;
109                 cumulativeDistribution_Sampler[i]=Cumul;
110         }
111         for(i=0; i<size; i++) cumulativeDistribution_Sampler[i]/=Cumul; // ↩
                normalize it
112         jumpTable_Sampler[0]=(i=0);
113         Cumul=uStep;
114         for(k=1;k<size;k++) {
115             while(cumulativeDistribution_Sampler[i]<=Cumul) i++;
116             jumpTable_Sampler[k]=i;
117             Cumul+=uStep;
118         }
119         return true;
120     }
121
122     protected boolean setTables_mixer(double[] inDist) {
123         int i; // true variable
124         int k; // uniform variable
125         double Cumul=0.; // For computing law cumulatives
126         double uStep=1./inDist.length; // uniform step
127         cumulativeDistribution_Mixer=new double[inDist.length];
128         jumpTable_Mixer=new int[inDist.length];
129         for(i=0; i<inDist.length; i++) { // compute the cumulative of inDist
130                 Cumul+=inDist[i];
131                 cumulativeDistribution_Mixer[i]=Cumul;
132         }
133         for(i=0; i<inDist.length; i++)
134                 cumulativeDistribution_Mixer[i]/=Cumul; // normalize it
135         jumpTable_Mixer[0]=(i=0);
136         Cumul=uStep;
137         for(k=1;k<inDist.length;k++) {
138             while(cumulativeDistribution_Mixer[i]<=Cumul) i++;
139             jumpTable_Mixer[k]=i;
140             Cumul+=uStep;
```

```
141         }
142         return true;
143     }
144 
145     protected int drawTable_sampler() {
146         if(__updated) if(!setTables_sampler()) return -1;
147         double x=rng.nextDouble();
148         int i=jumpTable_Sampler[(int) (jumpTable_Sampler.length*x)];
149         while (x>=cumulativeDistribution_Sampler[i]) i++;
150         return i;
151     }
152 
153     protected int drawTable_mixer() {
154         double x=rng.nextDouble();
155         int i=jumpTable_Mixer[(int) (jumpTable_Mixer.length*x)];
156         while (x>=cumulativeDistribution_Mixer[i]) i++;
157         return i;
158     }
159 
160 
161     ////////////////////////////////////////////////:
162     // public part
163     ////////////////
164 
165     @Override
166     public void update_notification() { // called each time a change is done
167         super.update_notification();
168         __updated=true;
169     }
170 
171     public double learnFrom(ArrayList<Assignment<Prop>> weightedSamples) {
172         int i,nbSamples;
173         double normalizer,aValue,Z;
174         Prop aProposition;
175         Assignment<Prop> anAssignment;
176         nbSamples=weightedSamples.size();
177         Z=(normalizer=0.);
178 
179         for(i=0;i<nbSamples;i++) {
180             anAssignment=weightedSamples.get(i);
181             aValue=anAssignment.value;
182             if(aValue<=0) {
183                 System.err.println("Error :: "+this.getClass().getName() +
184                                    ".learnFrom(ArrayList<Assignment<Prop>>)");
185                 System.err.println("Negative weight encountered");
186                 System.exit(0);
187             }
188             aProposition = anAssignment.attribute;
189             if(theZero==null) theZero = aProposition.instanceNsize().zero();
190             if(aProposition.compareTo(theZero)==0) Z+=aValue;
191             else normalizer+=aValue;
192         }
193         clear();
194         for(i=0;i<nbSamples;i++) {
195             anAssignment=weightedSamples.get(i);
196             aValue=anAssignment.value/normalizer;
197             aProposition = anAssignment.attribute;
198             if(aProposition.compareTo(theZero)!=0) add(aProposition,aValue);
199         }
200         return Z/(Z+normalizer); // return the percentage of reject
201     }
202 
203     public Assignment<Prop> makeSample() {
204         Assignment<Prop> assignment = new Assignment<Prop>();
205         assignment.value=1.;
206         assignment.attribute=mappedAttribute.get(drawTable_sampler()).attribute;
207         return assignment;
208     }
209 
210     public boolean setFuser(sB left, sB right,
211                             RefereeFunctionDefault<Prop> theRefereeFunction) {
212         ArrayList<sB> bbaTab= new ArrayList<sB>();
213         bbaTab.add(left);
214         bbaTab.add(right);
```

```
215         return setFuser(bbaTab,theRefereeFunction);
216     }
217 
218     public boolean setFuser(ArrayList<sB> bbaIn,
219                             RefereeFunctionDefault<Prop> aRefereeFunction) {
220         int i;
221         theRefereeFunction=aRefereeFunction;
222         fuserSources=bbaIn;
223         BBASources.clear();
224         for(i=0;i<fuserSources.size();i++) BBASources.add(fuserSources.get(i));
225 
226         for(i=0;i<bbaIn.size();i++) {
227             if(!bbaIn.get(i).setTables_sampler()) return false;
228         }
229         answeredAttribute=new ArrayList<Assignment<Prop>>();
230         for(i=0;i<bbaIn.size();i++) answeredAttribute.add(null);
231         return true;
232     }
233 
234     public Assignment<Prop> makeFusedSample() {
235         Assignment<Prop> assignment = new Assignment<Prop>();
236         Assignment<Prop> assignmentFromSource;
237         ArrayList<Assignment<Prop>> fusedAssignment;
238         int i;
239         double cumul=0.;
240         double x=rng.nextDouble();
241         assignment.value=1.;
242         for(i=0;i<fuserSources.size();i++) {
243             sB aSource=fuserSources.get(i);
244             assignmentFromSource=aSource.mappedAttribute.get(aSource.↩
                    drawTable_sampler());
245             answeredAttribute.set(i, assignmentFromSource);
246         }
247         fusedAssignment=theRefereeFunction.refereeFunction(answeredAttribute,↩
                BBASources);
248         for(i=0;i<fusedAssignment.size();i++) {
249             cumul+=fusedAssignment.get(i).value;
250             if(cumul>x) {
251                 assignment.attribute=fusedAssignment.get(i).attribute;
252                 return assignment;
253             }
254         }
255         return null;
256     }
257 
258     public boolean setMixer(double[] weights, ArrayList<sB> bbaIn) {
259         if(weights.length!=bbaIn.size()) return false;
260         mixerSources=bbaIn;
261         return setTables_mixer(weights);
262     }
263 
264     public sB makeMixedChoice() {
265         if(cumulativeDistribution_Mixer==null) return null;
266         int choice = drawTable_mixer();
267         return mixerSources.get(choice);
268     }
269 }
```

Source Code: RefereeToolbox_Tutorial.java

```
1  /*
2   *    RefereeToolbox_Tutorial.java : part of package RefereeToolbox; Tutorial.
3   ****************************************************************************
4   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
5   *
6   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
7   *
8   *    This file is part of RefereeToolbox.
9   *
10  *    RefereeToolbox is free software: you can redistribute it and/or modify
11  *    it under the terms of the GNU General Public License as published by
12  *    the Free Software Foundation, either version 3 of the License, or
13  *    (at your option) any later version.
14  *
15  *    RefereeToolbox is distributed in the hope that it will be useful,
16  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
17  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18  *    GNU General Public License for more details.
19  *
20  *    You should have received a copy of the GNU General Public License
21  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
22  ****************************************************************************
23  */
24
25 package RefereeToolbox;
26
27 // juste pour tests
28 import java.util.*;
29
30 /**
31  * Class providing a tutorial for package RefereeToolbox.
32  *
33  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
34  *
35  *
36  *
37  * <BR><BR><table border='1' cellPadding='4'><tr><td>
38  * <font color="#008000" style="font-family: georgia">
39  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
40  * <BR>
41  *    RefereeToolbox is free software: you can redistribute it and/or modify
42  *    it under the terms of the GNU General Public License as published by
43  *    the Free Software Foundation, either version 3 of the License, or
44  *    (at your option) any later version.<BR>
45  * <BR>
46  *    RefereeToolbox is distributed in the hope that it will be useful,
47  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
48  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
49  *    GNU General Public License for more details.<BR>
50  * <BR>
51  *    You should have received a copy of the GNU General Public License
52  *    along with RefereeToolbox. If not, see
53  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
54  * </font></table><BR>
55  *
56  */
57 public class RefereeToolbox_Tutorial{
58
59     /**
60      * Compare different referee functions applied with RefereeFuser and Powerset.
61      *
62      * <BR><BR>
63      * <b>Detailed code description.</b><BR>
64      * The following typonomic conventions are used:<BR>
65      * <code>
66      * <font color="#0000FF" style="font-family: georgia">
67      * Blue color is used for commenting the following code.
68      * </font><BR>
69      * code formating is used for printing the code.<BR>
70      * <font color="#FF0000" style="font-family: georgia">
71      * Red color is used for printing the possible output resulting of the
```

```
72      * previous code.
73      * </font>
74      * </code><BR>
75      * <BR><BR>
76      * Commented code:
77      * <BR><BR>
78      * <code>
79      * <font color="#0000FF" style="font-family: georgia">
80      * Declaration of propositon <i>A</i> as class <i>finalPowerset</i> and its
81      * creation;
82      * </font><BR>
83      * finalPowerset A=new finalPowerset();<BR>
84      * <font color="#0000FF" style="font-family: georgia">
85      * Set the size of the powerset to <i>3</i>; Then set <i>A</i> to the atomic
86      * proposition indiced by <i>0</i>;
87      * </font><BR>
88      * A.size(3); A.atomic(0);<BR>
89      * <font color="#0000FF" style="font-family: georgia">
90      * Declaration of <i>B</i> as <i>finalPowerset</i> and its creation and sizing
91      * as instance of <i>A</i>; Then set <i>B</i> to the atomic proposition ↩
            indiced
92      * by <i>1</i>;
93      * </font><BR>
94      * finalPowerset B=A.instanceNsize(); B.atomic(1);<BR>
95      * <font color="#0000FF" style="font-family: georgia">
96      * Declaration of <i>C</i> as <i>finalPowerset</i> and its creation and sizing
97      * as instance of <i>A</i>; Then set <i>C</i> to the atomic proposition ↩
            indiced
98      * by <i>2</i>;
99      * </font><BR>
100     * finalPowerset C=A.instanceNsize(); C.atomic(2);<BR>
101     * //<BR>
102     * <font color="#0000FF" style="font-family: georgia">
103     * Declaration of <i>AUB</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
104     * as instance of <i>A</i>; Then set <i>AUB</i> to <i>A OR B</i>;
105     * </font><BR>
106     * finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);<BR>
107     * <font color="#0000FF" style="font-family: georgia">
108     * Declaration of <i>BUC</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
109     * as instance of <i>A</i>; Then set <i>BUC</i> to <i>B OR C</i>;
110     * <BR>
111     * Declaration of <i>CUA</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
112     * as instance of <i>A</i>; Then set <i>CUA</i> to <i>C OR A</i>;
113     * </font><BR>
114     * finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);<BR>
115     * finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);<BR>
116     * //<BR>
117     * <font color="#0000FF" style="font-family: georgia">
118     * Declaration of <i>zero</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
119     * as instance of <i>A</i>; Then set <i>zero</i> to zero;
120     * <BR>
121     * Declaration of <i>one</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
122     * as instance of <i>A</i>; Then set <i>one</i> to one;
123     * </font><BR>
124     * finalPowerset zero=A.instanceNsize(); zero.zero();<BR>
125     * finalPowerset one=A.instanceNsize(); one.one();<BR>
126     * //<BR>
127     * //<BR>
128     * <font color="#0000FF" style="font-family: georgia">
129     * Declaration and creation of the referee function <i>referee1</i> as a ↩
            Dempster
130     * Shafer referee function on powerset;
131     * </font><BR>
132     * RFDempster_Powerset referee1 = new RFDempster_Powerset();<BR>
133     * <font color="#0000FF" style="font-family: georgia">
134     * Declaration and creation of the referee function <i>referee2</i> as a ↩
            Disjunctive
135     * referee function on powerset;<BR>
136     * Declaration and creation of the referee function <i>referee3</i> as a
```

```
137         * Dubois & Prade referee function on powerset;<BR>
138         * Declaration and creation of the referee function <i>referee4</i> as a PCR6
139         * referee function on powerset;<BR>
140         * Declaration and creation of the referee function <i>referee5</i> as a PCR#
141         * referee function on powerset;
142         * </font><BR>
143         * RFDisjunctive_Powerset referee2 = new RFDisjunctive_Powerset();<BR>
144         * RFDuboisPrade_Powerset referee3 = new RFDuboisPrade_Powerset();<BR>
145         * RFPCR6_Powerset referee4 = new RFPCR6_Powerset();<BR>
146         * RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();<BR>
147         * //<BR>
148         * //<BR>
149         * <font color="#0000FF" style="font-family: georgia">
150         * Declaration and creation of the basic belief assignment <i>aFuser1</i>, a
151         * bba defined over powerset and with the capability to handle fusion by means
152         * of referee functions (class {@link finalRefereeFuserRTS_Powerset});
153         * </font><BR>
154         * finalRefereeFuserRTS_Powerset aFuser1 = new finalRefereeFuserRTS_Powerset()↩
                ;<BR>
155         * <font color="#0000FF" style="font-family: georgia">
156         * Add assignment <i>(A,0.09)</i> to bba <i>aFuser1</i>;
157         * </font><BR>
158         * aFuser1.add(A,0.09);<BR>
159         * <font color="#0000FF" style="font-family: georgia">
160         * Add assignment <i>(B,0.2)</i> to bba <i>aFuser1</i>;<BR>
161         * Add assignment <i>(C,0.02)</i> to bba <i>aFuser1</i>;<BR>
162         * Add assignment <i>(AUB,0.05)</i> to bba <i>aFuser1</i>;<BR>
163         * Add assignment <i>(BUC,0.03)</i> to bba <i>aFuser1</i>;<BR>
164         * Add assignment <i>(CUA,0.1)</i> to bba <i>aFuser1</i>;
165         * </font><BR>
166         * aFuser1.add(B,0.2);<BR>
167         * aFuser1.add(C,0.02);<BR>
168         * aFuser1.add(AUB,0.05);<BR>
169         * aFuser1.add(BUC,0.03);<BR>
170         * aFuser1.add(CUA,0.1);<BR>
171         * <font color="#0000FF" style="font-family: georgia">
172         * Add assignment <i>(A,0.11)</i> to bba <i>aFuser1</i>. This assignment is
173         * <i>added</i> to previewsly added assignment <i>(A,0.09)</i>, resulting in
174         * total assignment <i>(A,0.2)</i>;
175         * </font><BR>
176         * aFuser1.add(A,0.11);<BR>
177         * <font color="#0000FF" style="font-family: georgia">
178         * Add assignment <i>(one,0.4)</i> to bba <i>aFuser1</i>;
179         * </font><BR>
180         * aFuser1.add(one,0.4);<BR>
181         * //<BR>
182         * <font color="#0000FF" style="font-family: georgia">
183         * Declaration and creation of the basic belief assignment <i>aFuser2</i>, a
184         * bba defined over powerset and with the capability to handle fusion by means
185         * of referee functions (class {@link finalRefereeFuserRTS_Powerset});
186         * </font><BR>
187         * finalRefereeFuserRTS_Powerset aFuser2 = aFuser1.instance();<BR>
188         * <font color="#0000FF" style="font-family: georgia">
189         * Add assignment <i>(A,0.1)</i> to bba <i>aFuser2</i>;<BR>
190         * Add assignment <i>(B,0.1)</i> to bba <i>aFuser2</i>;<BR>
191         * Add assignment <i>(C,0.2)</i> to bba <i>aFuser2</i>;<BR>
192         * Add assignment <i>(AUB,0.2)</i> to bba <i>aFuser2</i>;<BR>
193         * Add assignment <i>(BUC,0.1)</i> to bba <i>aFuser2</i>;<BR>
194         * Add assignment <i>(CUA,0.1)</i> to bba <i>aFuser2</i>;<BR>
195         * Add assignment <i>(one,0.2)</i> to bba <i>aFuser2</i>;
196         * </font><BR>
197         * aFuser2.add(A,0.1);<BR>
198         * aFuser2.add(B,0.1);<BR>
199         * aFuser2.add(C,0.2);<BR>
200         * aFuser2.add(AUB,0.2);<BR>
201         * aFuser2.add(BUC,0.1);<BR>
202         * aFuser2.add(CUA,0.1);<BR>
203         * aFuser2.add(one,0.2);<BR>
204         * //<BR>
205         * <font color="#0000FF" style="font-family: georgia">
206         * Declaration and creation of the basic belief assignment <i>aFuser</i>, a
207         * bba defined over powerset and with the capability to handle fusion by means
208         * of referee functions (class {@link finalRefereeFuserRTS_Powerset});
209         * </font><BR>
```

```
210       * finalRefereeFuserRTS_Powerset aFuser = aFuser1.instance();<BR>
211       * //<BR>
212       * <font color="#0000FF" style="font-family: georgia">
213       * Define a common print mode <i>printMode</i> for the bba state.
214       * This variable, set to <i>1</i>, implies the print to be according to the
215       * proposition order;
216       * </font><BR>
217       * int printMode=1;<BR>
218       * <font color="#0000FF" style="font-family: georgia">
219       * Print the title of the method;
220       * </font><BR>
221       * System.out.println(<BR>
222       *        
223       * "////////////////////////////\n" +<BR>
224       *        
225       * "// Relaxed Method\n" +<BR>
226       *        
227       * "////////////////////\n");<BR>
228       * <font color="#FF0000" style="font-family: georgia">
229       * ////////////////////////////<BR>
230       * // Relaxed Method<BR>
231       * ////////////////////
232       * </font><BR>
233       * System.out.println("aFuser1");<BR>
234       * <font color="#FF0000" style="font-family: georgia">
235       * aFuser1
236       * </font><BR>
237       * <font color="#0000FF" style="font-family: georgia">
238       * Print the state of <i>aFuser1</i>. With option <i>printMode=1</i>, the ↩
              print is
239       * done according to the proposition order;
240       * </font><BR>
241       *    
242       * System.out.println(aFuser1.state(printMode));<BR>
243       * <font color="#FF0000" style="font-family: georgia">
244       * bbaTreeA:<BR>
245       * x0000000000000001 -> 0.2<BR>
246       * x0000000000000002 -> 0.2<BR>
247       * x0000000000000003 -> 0.05<BR>
248       * x0000000000000004 -> 0.02<BR>
249       * x0000000000000005 -> 0.1<BR>
250       * x0000000000000006 -> 0.03<BR>
251       * x0000000000000007 -> 0.4
252       * </font><BR>
253       *    
254       * System.out.println();<BR>
255       * <font color="#0000FF" style="font-family: georgia">
256       * Print the state of <i>aFuser2</i>. With option <i>printMode=1</i>, the ↩
              print is
257       * done according to the proposition order;
258       * </font><BR>
259       * System.out.println("aFuser2");<BR>
260       *    
261       * System.out.println(aFuser2.state(printMode));<BR>
262       *    
263       * System.out.println();<BR>
264       * <font color="#FF0000" style="font-family: georgia">
265       * aFuser2<BR>
266       * bbaTreeA:<BR>
267       * x0000000000000001 -> 0.1<BR>
268       * x0000000000000002 -> 0.1<BR>
269       * x0000000000000003 -> 0.2<BR>
270       * x0000000000000004 -> 0.2<BR>
271       * x0000000000000005 -> 0.1<BR>
272       * x0000000000000006 -> 0.1<BR>
273       * x0000000000000007 -> 0.2
274       * </font><BR>
275       * <font color="#0000FF" style="font-family: georgia">
276       * Set <i>aFuser</i> as the fusion of <i>aFuser1</i> and <i>aFuser2</i>, ↩
              according
277       * to the referee function <i>referee1</i>. As a result, a Dempster Shafer
278       * combination is done;
279       * </font><BR>
280       * aFuser.fuse(aFuser1,aFuser2,referee1);<BR>
```

```
281         * <font color="#0000FF" style="font-family: georgia">
282         * Print the state of <i>aFuser</i>. With option <i>printMode=1</i>, the print↩
                is
283         * done according to the proposition order;
284         * </font><BR>
285         *    
286         * System.out.println("aFuser - Dempster");<BR>
287         *     System.out.println("Conflict Z = "+aFuser.conflict()+" %");
288         *    
289         * System.out.println(aFuser.state(printMode));<BR>
290         *    
291         * System.out.println();<BR>
292         * <font color="#FF0000" style="font-family: georgia">
293         * aFuser - Dempster<BR>
294         * Conflict Z = 0.19100000000000006 %
295         * bbaTreeA:<BR>
296         * x0000000000000001 -> 0.24721878862793575<BR>
297         * x0000000000000002 -> 0.2212608158220025<BR>
298         * x0000000000000003 -> 0.12360939431396788<BR>
299         * x0000000000000004 -> 0.16192830655129792<BR>
300         * x0000000000000005 -> 0.08652657601977749<BR>
301         * x0000000000000006 -> 0.06056860321384425<BR>
302         * x0000000000000007 -> 0.09888751545117429
303         * </font><BR>
304         * <font color="#0000FF" style="font-family: georgia">
305         * Set <i>aFuser</i> as the fusion of <i>aFuser1</i> and <i>aFuser2</i>, ↩
              according
306         * to the referee function <i>referee2</i>. As a result, a disjunctive
307         * combination is done;
308         * </font><BR>
309         * aFuser.fuse(aFuser1,aFuser2,referee2);<BR>
310         * <font color="#0000FF" style="font-family: georgia">
311         * Print the state of <i>aFuser</i>. With option <i>printMode=1</i>, the print↩
                is
312         * done according to the proposition order;
313         * </font><BR>
314         *    
315         * System.out.println("aFuser - Disjunctive");<BR>
316         *     System.out.println("Conflict Z = "+aFuser.conflict()+" %");
317         *    
318         * System.out.println(aFuser.state(printMode));<BR>
319         *    
320         * System.out.println();<BR>
321         * <font color="#FF0000" style="font-family: georgia">
322         * aFuser - Disjunctive<BR>
323         * Conflict Z = 0.0 %
324         * bbaTreeA:<BR>
325         * x0000000000000001 -> 0.019999999999999993<BR>
326         * x0000000000000002 -> 0.019999999999999993<BR>
327         * x0000000000000003 -> 0.13999999999999999<BR>
328         * x0000000000000004 -> 0.003999999999999998<BR>
329         * x0000000000000005 -> 0.10399999999999997<BR>
330         * x0000000000000006 -> 0.07599999999999998<BR>
331         * x0000000000000007 -> 0.636
332         * </font><BR>
333         * <font color="#0000FF" style="font-family: georgia">
334         * Set <i>aFuser</i> as the fusion of <i>aFuser1</i> and <i>aFuser2</i>, ↩
              according
335         * to the referee function <i>referee3</i>. As a result, a Dubois & Prade
336         * combination is done;
337         * </font><BR>
338         * aFuser.fuse(aFuser1,aFuser2,referee3);<BR>
339         * <font color="#0000FF" style="font-family: georgia">
340         * Print the state of <i>aFuser</i>. With option <i>printMode=1</i>, the print↩
                is
341         * done according to the proposition order;
342         * </font><BR>
343         *    
344         * System.out.println("aFuser - Dubois & Prade");<BR>
345         *     System.out.println("Conflict Z = "+aFuser.conflict()+" %");
346         *    
347         * System.out.println(aFuser.state(printMode));<BR>
348         *    
349         * System.out.println();<BR>
```

```
350         * <font color="#FF0000" style="font-family: georgia">
351         * aFuser - Dubois & Prade<BR>
352         * Conflict Z = 0.0 %
353         * bbaTreeA:<BR>
354         * x0000000000000001 -> 0.2<BR>
355         * x0000000000000002 -> 0.17900000000000002<BR>
356         * x0000000000000003 -> 0.14<BR>
357         * x0000000000000004 -> 0.131<BR>
358         * x0000000000000005 -> 0.11199999999999999<BR>
359         * x0000000000000006 -> 0.09100000000000001<BR>
360         * x0000000000000007 -> 0.147
361         * </font><BR>
362         * <font color="#0000FF" style="font-family: georgia">
363         * Set <i>aFuser</i> as the fusion of <i>aFuser1</i> and <i>aFuser2</i>, ↩
              according
364         * to the referee function <i>referee4</i>. As a result, a PCR6 combination
365         * is done;
366         * </font><BR>
367         * aFuser.fuse(aFuser1,aFuser2,referee4);<BR>
368         * <font color="#0000FF" style="font-family: georgia">
369         * Print the state of <i>aFuser</i>. With option <i>printMode=1</i>, the print↩
              is
370         * done according to the proposition order;
371         * </font><BR>
372         *    
373         * System.out.println("aFuser - PCR6");<BR>
374         *     System.out.println("Conflict Z = "+aFuser.conflict()+" %");
375         *    
376         * System.out.println(aFuser.state(printMode));<BR>
377         *    
378         * System.out.println();<BR>
379         * <font color="#FF0000" style="font-family: georgia">
380         * aFuser - PCR6<BR>
381         * Conflict Z = 0.0 %
382         * bbaTreeA:<BR>
383         * x0000000000000001 -> 0.2573076923076923<BR>
384         * x0000000000000002 -> 0.23900000000000005<BR>
385         * x0000000000000003 -> 0.10563636363636364<BR>
386         * x0000000000000004 -> 0.18003030303030304<BR>
387         * x0000000000000005 -> 0.08166666666666668<BR>
388         * x0000000000000006 -> 0.056358974358974356<BR>
389         * x0000000000000007 -> 0.08
390         * </font><BR>
391         * <font color="#0000FF" style="font-family: georgia">
392         * Set <i>aFuser</i> as the fusion of <i>aFuser1</i> and <i>aFuser2</i>, ↩
              according
393         * to the referee function <i>referee5</i>. As a result, a PCR# combination
394         * is done;
395         * </font><BR>
396         * aFuser.fuse(aFuser1,aFuser2,referee5);<BR>
397         * <font color="#0000FF" style="font-family: georgia">
398         * Print the state of <i>aFuser</i>. With option <i>printMode=1</i>, the print↩
              is
399         * done according to the proposition order;
400         * </font><BR>
401         *    
402         * System.out.println("aFuser - PCR#");<BR>
403         *     System.out.println("Conflict Z = "+aFuser.conflict()+" %");
404         *    
405         * System.out.println(aFuser.state(printMode));<BR>
406         *    
407         * System.out.println();<BR>
408         * <font color="#FF0000" style="font-family: georgia">
409         * aFuser - PCR#<BR>
410         * Conflict Z = 0.0 %
411         * bbaTreeA:<BR>
412         * x0000000000000001 -> 0.2573076923076923<BR>
413         * x0000000000000002 -> 0.23900000000000005<BR>
414         * x0000000000000003 -> 0.10563636363636364<BR>
415         * x0000000000000004 -> 0.18003030303030304<BR>
416         * x0000000000000005 -> 0.08166666666666668<BR>
417         * x0000000000000006 -> 0.056358974358974356<BR>
418         * x0000000000000007 -> 0.08
419         * </font><BR>
```

```
420         * <BR>
421         * </code>
422         *
423         */
424        public void RefereeFuserRTS_Comparison() {
425            finalPowerset A=new finalPowerset();
426            A.size(3); A.atomic(0);
427            finalPowerset B=A.instanceNsize(); B.atomic(1);
428            finalPowerset C=A.instanceNsize(); C.atomic(2);
429            //
430            finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);
431            finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);
432            finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);
433            //
434            finalPowerset zero=A.instanceNsize(); zero.zero();
435            finalPowerset one=A.instanceNsize(); one.one();
436            //
437            //
438            RFDempster_Powerset referee1 = new RFDempster_Powerset();
439            RFDisjunctive_Powerset referee2 = new RFDisjunctive_Powerset();
440            RFDuboisPrade_Powerset referee3 = new RFDuboisPrade_Powerset();
441            RFPCR6_Powerset referee4 = new RFPCR6_Powerset();
442            RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();
443            //
444            //
445            finalRefereeFuserRTS_Powerset aFuser1 = new finalRefereeFuserRTS_Powerset↩
                   ();
446            aFuser1.add(A,0.09);
447            aFuser1.add(B,0.2);
448            aFuser1.add(C,0.02);
449            aFuser1.add(AUB,0.05);
450            aFuser1.add(BUC,0.03);
451            aFuser1.add(CUA,0.1);
452            aFuser1.add(A,0.11);
453            aFuser1.add(one,0.4);
454            //
455            finalRefereeFuserRTS_Powerset aFuser2 = aFuser1.instance();
456            aFuser2.add(A,0.1);
457            aFuser2.add(B,0.1);
458            aFuser2.add(C,0.2);
459            aFuser2.add(AUB,0.2);
460            aFuser2.add(BUC,0.1);
461            aFuser2.add(CUA,0.1);
462            aFuser2.add(one,0.2);
463            //
464            finalRefereeFuserRTS_Powerset aFuser = aFuser1.instance();
465    //
466
467            int printMode=1;
468
469            System.out.println(
470                    "////////////////////////////\n" +
471                    "// Relaxed Method\n" +
472                    "////////////////////\n");
473
474            System.out.println("aFuser1");
475                System.out.println(aFuser1.state(printMode));
476                System.out.println();
477            System.out.println("aFuser2");
478                System.out.println(aFuser2.state(printMode));
479                System.out.println();
480            aFuser.fuse(aFuser1,aFuser2,referee1);
481            System.out.println("aFuser - Dempster");
482                System.out.println("Conflict Z = "+aFuser.conflict()+" %");
483                System.out.println(aFuser.state(printMode));
484                System.out.println();
485            aFuser.fuse(aFuser1,aFuser2,referee2);
486            System.out.println("aFuser - Disjunctive");
487                System.out.println("Conflict Z = "+aFuser.conflict()+" %");
488                System.out.println(aFuser.state(printMode));
489                System.out.println();
490            aFuser.fuse(aFuser1,aFuser2,referee3);
491            System.out.println("aFuser - Dubois & Prade");
492                System.out.println("Conflict Z = "+aFuser.conflict()+" %");
```

```
493             System.out.println(aFuser.state(printMode));
494             System.out.println();
495         aFuser.fuse(aFuser1,aFuser2,referee4);
496         System.out.println("aFuser - PCR6");
497             System.out.println("Conflict Z = "+aFuser.conflict()+" %");
498             System.out.println(aFuser.state(printMode));
499             System.out.println();
500         aFuser.fuse(aFuser1,aFuser2,referee5);
501         System.out.println("aFuser - PCR#");
502             System.out.println("Conflict Z = "+aFuser.conflict()+" %");
503             System.out.println(aFuser.state(printMode));
504             System.out.println();
505
506     }
507
508     /**
509      * Compare different referee functions applied with RefereeSampler and ↩
            Powerset.
510      *
511      * <BR><BR>
512      * <b>Detailed code description.</b><BR>
513      * The following typonomic conventions are used:<BR>
514      * <code>
515      * <font color="#0000FF" style="font-family: georgia">
516      * Blue color is used for commenting the following code.
517      * </font><BR>
518      * code formating is used for printing the code.<BR>
519      * <font color="#FF0000" style="font-family: georgia">
520      * Red color is used for printing the possible output resulting of the
521      * previous code.
522      * </font>
523      * </code><BR>
524      * <BR><BR>
525      * Commented code:
526      * <BR><BR>
527      * <code>
528      * <font color="#0000FF" style="font-family: georgia">
529      * Declaration of propositon <i>A</i> as class <i>finalPowerset</i> and its
530      * creation;
531      * </font><BR>
532      * finalPowerset A=new finalPowerset();<BR>
533      * <font color="#0000FF" style="font-family: georgia">
534      * Set the size of the powerset to <i>3</i>; Then set <i>A</i> to the atomic
535      * proposition indiced by <i>0</i>;
536      * </font><BR>
537      * A.size(3); A.atomic(0);<BR>
538      * <font color="#0000FF" style="font-family: georgia">
539      * Declaration of <i>B</i> as <i>finalPowerset</i> and its creation and sizing
540      * as instance of <i>A</i>; Then set <i>B</i> to the atomic proposition ↩
            indiced
541      * by <i>1</i>;
542      * </font><BR>
543      * finalPowerset B=A.instanceNsize(); B.atomic(1);<BR>
544      * <font color="#0000FF" style="font-family: georgia">
545      * Declaration of <i>C</i> as <i>finalPowerset</i> and its creation and sizing
546      * as instance of <i>A</i>; Then set <i>C</i> to the atomic proposition ↩
            indiced
547      * by <i>2</i>;
548      * </font><BR>
549      * finalPowerset C=A.instanceNsize(); C.atomic(2);<BR>
550      * //<BR>
551      * <font color="#0000FF" style="font-family: georgia">
552      * Declaration of <i>AUB</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
553      * as instance of <i>A</i>; Then set <i>AUB</i> to <i>A OR B</i>;
554      * </font><BR>
555      * finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);<BR>
556      * <font color="#0000FF" style="font-family: georgia">
557      * Declaration of <i>BUC</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
558      * as instance of <i>A</i>; Then set <i>BUC</i> to <i>B OR C</i>;
559      * <BR>
560      * Declaration of <i>CUA</i> as <i>finalPowerset</i> and its creation and ↩
            sizing
```

```
561       * as instance of <i>A</i>; Then set <i>CUA</i> to <i>C OR A</i>;
562       * </font><BR>
563       * finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);<BR>
564       * finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);<BR>
565       * //<BR>
566       * <font color="#0000FF" style="font-family: georgia">
567       * Declaration of <i>zero</i> as <i>finalPowerset</i> and its creation and ↩
              sizing
568       * as instance of <i>A</i>; Then set <i>zero</i> to zero;
569       * <BR>
570       * Declaration of <i>one</i> as <i>finalPowerset</i> and its creation and ↩
              sizing
571       * as instance of <i>A</i>; Then set <i>one</i> to one;
572       * </font><BR>
573       * finalPowerset zero=A.instanceNsize(); zero.zero();<BR>
574       * finalPowerset one=A.instanceNsize(); one.one();<BR>
575       * //<BR>
576       * //<BR>
577       * <font color="#0000FF" style="font-family: georgia">
578       * Declaration and creation of the referee function <i>referee1</i> as a ↩
              Dempster
579       * Shafer referee function on powerset;
580       * </font><BR>
581       * RFDempster_Powerset referee1 = new RFDempster_Powerset();<BR>
582       * <font color="#0000FF" style="font-family: georgia">
583       * Declaration and creation of the referee function <i>referee2</i> as a ↩
              Disjunctive
584       * referee function on powerset;<BR>
585       * Declaration and creation of the referee function <i>referee3</i> as a
586       * Dubois & Prade referee function on powerset;<BR>
587       * Declaration and creation of the referee function <i>referee4</i> as a PCR6
588       * referee function on powerset;<BR>
589       * Declaration and creation of the referee function <i>referee5</i> as a PCR#
590       * referee function on powerset;
591       * </font><BR>
592       * RFDisjunctive_Powerset referee2 = new RFDisjunctive_Powerset();<BR>
593       * RFDuboisPrade_Powerset referee3 = new RFDuboisPrade_Powerset();<BR>
594       * RFPCR6_Powerset referee4 = new RFPCR6_Powerset();<BR>
595       * RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();<BR>
596       * //<BR>
597       * //<BR>
598       * <font color="#0000FF" style="font-family: georgia">
599       * Declaration and creation of the basic belief assignment <i>aSampler1</i>, a
600       * bba defined over powerset and with the capability to handle fusion by means
601       * of referee sampling (class {@link finalRefereeSampler_Powerset});
602       * </font><BR>
603       * finalRefereeSampler_Powerset aSampler1 = new finalRefereeSampler_Powerset()↩
              ;<BR>
604       * <font color="#0000FF" style="font-family: georgia">
605       * Add assignment <i>(A,0.09)</i> to bba <i>aSampler1</i>;
606       * </font><BR>
607       * aSampler1.add(A,0.09);<BR>
608       * <font color="#0000FF" style="font-family: georgia">
609       * Add assignment <i>(B,0.2)</i> to bba <i>aSampler1</i>;<BR>
610       * Add assignment <i>(C,0.02)</i> to bba <i>aSampler1</i>;<BR>
611       * Add assignment <i>(AUB,0.05)</i> to bba <i>aSampler1</i>;<BR>
612       * Add assignment <i>(BUC,0.03)</i> to bba <i>aSampler1</i>;<BR>
613       * Add assignment <i>(CUA,0.1)</i> to bba <i>aSampler1</i>;
614       * </font><BR>
615       * aSampler1.add(B,0.2);<BR>
616       * aSampler1.add(C,0.02);<BR>
617       * aSampler1.add(AUB,0.05);<BR>
618       * aSampler1.add(BUC,0.03);<BR>
619       * aSampler1.add(CUA,0.1);<BR>
620       * <font color="#0000FF" style="font-family: georgia">
621       * Add assignment <i>(A,0.11)</i> to bba <i>aSampler1</i>. This assignment is
622       * <i>added</i> to previewsly added assignment <i>(A,0.09)</i>, resulting in
623       * total assignment <i>(A,0.2)</i>;
624       * </font><BR>
625       * aSampler1.add(A,0.11);<BR>
626       * <font color="#0000FF" style="font-family: georgia">
627       * Add assignment <i>(one,0.4)</i> to bba <i>aSampler1</i>;
628       * </font><BR>
629       * aSampler1.add(one,0.4);<BR>
```

```
630         * //<BR>
631         * <font color="#0000FF" style="font-family: georgia">
632         * Declaration and creation of the basic belief assignment <i>aSampler2</i>, a
633         * bba defined over powerset and with the capability to handle fusion by means
634         * of referee sampling (class {@link finalRefereeSampler_Powerset});
635         * </font><BR>
636         * finalRefereeSampler_Powerset aSampler2 = aSampler1.instance();<BR>
637         * <font color="#0000FF" style="font-family: georgia">
638         * Add assignment <i>(A,0.1)</i> to bba <i>aSampler2</i>;<BR>
639         * Add assignment <i>(B,0.1)</i> to bba <i>aSampler2</i>;<BR>
640         * Add assignment <i>(C,0.2)</i> to bba <i>aSampler2</i>;<BR>
641         * Add assignment <i>(AUB,0.2)</i> to bba <i>aSampler2</i>;<BR>
642         * Add assignment <i>(BUC,0.1)</i> to bba <i>aSampler2</i>;<BR>
643         * Add assignment <i>(CUA,0.1)</i> to bba <i>aSampler2</i>;<BR>
644         * Add assignment <i>(one,0.2)</i> to bba <i>aSampler2</i>;
645         * </font><BR>
646         * aSampler2.add(A,0.1);<BR>
647         * aSampler2.add(B,0.1);<BR>
648         * aSampler2.add(C,0.2);<BR>
649         * aSampler2.add(AUB,0.2);<BR>
650         * aSampler2.add(BUC,0.1);<BR>
651         * aSampler2.add(CUA,0.1);<BR>
652         * aSampler2.add(one,0.2);<BR>
653         * //<BR>
654         * <font color="#0000FF" style="font-family: georgia">
655         * Declaration and creation of the basic belief assignment <i>aSampler</i>, a
656         * bba defined over powerset and with the capability to handle fusion by means
657         * of referee sampling (class {@link finalRefereeSampler_Powerset});
658         * </font><BR>
659         * finalRefereeSampler_Powerset aSampler = aSampler1.instance();<BR>
660         * //<BR>
661         * <font color="#0000FF" style="font-family: georgia">
662         * Define a common print mode <i>printMode</i> for the bba state.
663         * This variable, set to <i>1</i>, implies the print to be according to the
664         * proposition order;
665         * </font><BR>
666         * int printMode=1;<BR>
667         * <font color="#0000FF" style="font-family: georgia">
668         * Print the title of the method;
669         * </font><BR>
670         * System.out.println(<BR>
671         *        
672         * "////////////////////////////////\n" +<BR>
673         *        
674         * "// Sampling Method\n" +<BR>
675         *        
676         * "////////////////////\n");<BR>
677         * <font color="#FF0000" style="font-family: georgia">
678         * ////////////////////////////////<BR>
679         * // Sampling Method<BR>
680         * ////////////////////
681         * </font><BR>
682         *
683         * <font color="#0000FF" style="font-family: georgia">
684         * Variables declaration.
685         * <i>n</i> and <i>Z</i> are instrumental and used for sample indices and ↩
              contradiction
686         * degree repectively.
687         * <i>NbSamples</i> stores the number of samples generated by each computation↩
              ;
688         * </font><BR>
689         * int n;<BR>
690         * int NbSamples=1000000;<BR>
691         * double Z;<BR>
692         *
693         * <font color="#0000FF" style="font-family: georgia">
694         * Declaration and creation of the basic belief assignment <i>Samples</i>. ↩
              This
695         * bba is instrumental and used for computing the samples statistic ↩
              incrementally;
696         * </font><BR>
697         * finalRefereeSampler_Powerset Samples= aSampler1.instance();<BR>
698         * <font color="#0000FF" style="font-family: georgia">
699         * Print the state of <i>aSampler1</i>. With option <i>printMode=1</i>, the ↩
```

```
        print is
700     * done according to the proposition order;
701     * </font><BR>
702     * System.out.println("aSampler1");<BR>
703     *     System.out.println(aSampler1.state(printMode));<BR>
704     * <font color="#FF0000" style="font-family: georgia">
705     * aSampler1<BR>
706     * bbaTreeA:<BR>
707     * x0000000000000001 -> 0.2<BR>
708     * x0000000000000002 -> 0.2<BR>
709     * x0000000000000003 -> 0.05<BR>
710     * x0000000000000004 -> 0.02<BR>
711     * x0000000000000005 -> 0.1<BR>
712     * x0000000000000006 -> 0.03<BR>
713     * x0000000000000007 -> 0.4
714     * </font><BR>
715     *     System.out.println();<BR>
716     *
717     * <font color="#0000FF" style="font-family: georgia">
718     * Print the state of <i>aSampler2</i>. With option <i>printMode=1</i>, the
719     * print is done according to the proposition order;
720     * </font><BR>
721     * System.out.println("aSampler2");<BR>
722     *     System.out.println(aSampler2.state(printMode));<BR>
723     *     System.out.println();<BR>
724     * //<BR>
725     * <font color="#FF0000" style="font-family: georgia">
726     * aSampler2<BR>
727     * bbaTreeA:<BR>
728     * x0000000000000001 -> 0.1<BR>
729     * x0000000000000002 -> 0.1<BR>
730     * x0000000000000003 -> 0.2<BR>
731     * x0000000000000004 -> 0.2<BR>
732     * x0000000000000005 -> 0.1<BR>
733     * x0000000000000006 -> 0.1<BR>
734     * x0000000000000007 -> 0.2
735     * </font><BR>
736     *
737     * <font color="#0000FF" style="font-family: georgia">
738     * Initialize the fuser of <i>aSampler</i> with bba enties <i>aFuser1</i> and
739     * <i>aFuser2</i>, and with the referee function <i>referee1</i> (Dempster ↩
            Shafer);
740     * </font><BR>
741     * aSampler.setFuser(aSampler1, aSampler2, referee1);<BR>
742     * <font color="#0000FF" style="font-family: georgia">
743     * Clear the samples container <i>Samples</i>;
744     * </font><BR>
745     * Samples.clear();<BR>
746     * <font color="#0000FF" style="font-family: georgia">
747     * Build the samples (<i>NbSamples</i> times) fused by means of the fuser and
748     * store them within the container;
749     * </font><BR>
750     * {@code for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());}<↩
            BR>
751     * <font color="#0000FF" style="font-family: georgia">
752     * Learn the bba <i>aSampler</i> from the samples <i>Samples</i> and store the
753     * percentage of conflict samples within <i>Z</i>;
754     * </font><BR>
755     * Z=aSampler.learnFrom(Samples.toArray());<BR>
756     * <font color="#0000FF" style="font-family: georgia">
757     * Print the conflict and the state of <i>aSampler</i> (these results are ↩
            random).
758     * With option <i>printMode=1</i>, the print is done according to the ↩
            proposition
759     * order;
760     * </font><BR>
761     * System.out.println("aSampler - Dempster -- " + NbSamples + " particles");<↩
            BR>
762     *     System.out.println("Conflict Z = "+Z+" %");<BR>
763     *     System.out.println(aSampler.state(printMode));<BR>
764     *     System.out.println();<BR>
765     *     //<BR>
766     * <font color="#FF0000" style="font-family: georgia">
767     * aSampler - Dempster -- 1000000 particles<BR>
```

```
768       * Conflict Z = 0.190523 %<BR>
769       * bbaTreeA:<BR>
770       * x0000000000000001 -> 0.2474844868970953<BR>
771       * x0000000000000002 -> 0.2219408334023079<BR>
772       * x0000000000000003 -> 0.1233846051215785<BR>
773       * x0000000000000004 -> 0.16099654468255428<BR>
774       * x0000000000000005 -> 0.08678813604339593<BR>
775       * x0000000000000006 -> 0.06061197538657676<BR>
776       * x0000000000000007 -> 0.09879341846649133<BR>
777       * </font><BR>
778       *
779       * <font color="#0000FF" style="font-family: georgia">
780       * Initialize the fuser of <i>aSampler</i> with bba enties <i>aFuser1</i> and
781       * <i>aFuser2</i>, and with the referee function <i>referee2</i> (Disjunctive)↩
              ;
782       * </font><BR>
783       * aSampler.setFuser(aSampler1, aSampler2, referee2);<BR>
784       * <font color="#0000FF" style="font-family: georgia">
785       * Clear the samples container <i>Samples</i>;
786       * </font><BR>
787       * Samples.clear();<BR>
788       * <font color="#0000FF" style="font-family: georgia">
789       * Build the samples (<i>NbSamples</i> times) fused by means of the fuser and
790       * store them within the container;
791       * </font><BR>
792       * {@code for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());}<↩
              BR>
793       * <font color="#0000FF" style="font-family: georgia">
794       * Learn the bba <i>aSampler</i> from the samples <i>Samples</i> and store the
795       * percentage of conflict samples within <i>Z</i>;
796       * </font><BR>
797       * Z=aSampler.learnFrom(Samples.toArray());<BR>
798       * <font color="#0000FF" style="font-family: georgia">
799       * Print the conflict and the state of <i>aSampler</i> (these results are ↩
              random).
800       * With option <i>printMode=1</i>, the print is done according to the ↩
              proposition
801       * order;
802       * </font><BR>
803       * System.out.println("aSampler - Disjunctive -- " + NbSamples + " particles")↩
              ;<BR>
804       *     System.out.println("Conflict Z = "+Z+" %");<BR>
805       *     System.out.println(aSampler.state(printMode));<BR>
806       *     System.out.println();<BR>
807       *     //<BR>
808       * <font color="#FF0000" style="font-family: georgia">
809       * aSampler - Disjunctive -- 1000000 particles<BR>
810       * Conflict Z = 0.0 %<BR>
811       * bbaTreeA:<BR>
812       * x0000000000000001 -> 0.01991<BR>
813       * x0000000000000002 -> 0.019988<BR>
814       * x0000000000000003 -> 0.139937<BR>
815       * x0000000000000004 -> 0.004052<BR>
816       * x0000000000000005 -> 0.103861<BR>
817       * x0000000000000006 -> 0.075948<BR>
818       * x0000000000000007 -> 0.636304
819       * </font><BR>
820       *
821       * <font color="#0000FF" style="font-family: georgia">
822       * Initialize the fuser of <i>aSampler</i> with bba enties <i>aFuser1</i> and
823       * <i>aFuser2</i>, and with the referee function <i>referee3</i> (Dubois & ↩
              Prade);
824       * </font><BR>
825       * aSampler.setFuser(aSampler1, aSampler2, referee3);<BR>
826       * <font color="#0000FF" style="font-family: georgia">
827       * Clear the samples container <i>Samples</i>;
828       * </font><BR>
829       * Samples.clear();<BR>
830       * <font color="#0000FF" style="font-family: georgia">
831       * Build the samples (<i>NbSamples</i> times) fused by means of the fuser and
832       * store them within the container;
833       * </font><BR>
834       * {@code for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());}<↩
              BR>
```

```
835         * <font color="#0000FF" style="font-family: georgia">
836         * Learn the bba <i>aSampler</i> from the samples <i>Samples</i> and store the
837         * percentage of conflict samples within <i>Z</i>;
838         * </font><BR>
839         * Z=aSampler.learnFrom(Samples.toArray());<BR>
840         * <font color="#0000FF" style="font-family: georgia">
841         * Print the conflict and the state of <i>aSampler</i> (these results are ↩
                random).
842         * With option <i>printMode=1</i>, the print is done according to the ↩
                proposition
843         * order;
844         * </font><BR>
845         * System.out.println("aSampler - Dubois & Prade -- " + NbSamples + " ↩
                particles");<BR>
846         *     System.out.println("Conflict Z = "+Z+" %");<BR>
847         *     System.out.println(aSampler.state(printMode));<BR>
848         *     System.out.println();<BR>
849         *     //<BR>
850         * <font color="#FF0000" style="font-family: georgia">
851         * aSampler - Dubois & Prade -- 1000000 particles<BR>
852         * Conflict Z = 0.0 %<BR>
853         * bbaTreeA:<BR>
854         * x0000000000000001 -> 0.200409<BR>
855         * x0000000000000002 -> 0.179814<BR>
856         * x0000000000000003 -> 0.139608<BR>
857         * x0000000000000004 -> 0.130784<BR>
858         * x0000000000000005 -> 0.111702<BR>
859         * x0000000000000006 -> 0.090511<BR>
860         * x0000000000000007 -> 0.147172
861         * </font><BR>
862         *
863         * <font color="#0000FF" style="font-family: georgia">
864         * Initialize the fuser of <i>aSampler</i> with bba enties <i>aFuser1</i> and
865         * <i>aFuser2</i>, and with the referee function <i>referee4</i> (PCR6);
866         * </font><BR>
867         * aSampler.setFuser(aSampler1, aSampler2, referee4);<BR>
868         * <font color="#0000FF" style="font-family: georgia">
869         * Clear the samples container <i>Samples</i>;
870         * </font><BR>
871         * Samples.clear();<BR>
872         * <font color="#0000FF" style="font-family: georgia">
873         * Build the samples (<i>NbSamples</i> times) fused by means of the fuser and
874         * store them within the container;
875         * </font><BR>
876         * {@code for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());}<↩
                BR>
877         * <font color="#0000FF" style="font-family: georgia">
878         * Learn the bba <i>aSampler</i> from the samples <i>Samples</i> and store the
879         * percentage of conflict samples within <i>Z</i>;
880         * </font><BR>
881         * Z=aSampler.learnFrom(Samples.toArray());<BR>
882         * <font color="#0000FF" style="font-family: georgia">
883         * Print the conflict and the state of <i>aSampler</i> (these results are ↩
                random).
884         * With option <i>printMode=1</i>, the print is done according to the ↩
                proposition
885         * order;
886         * </font><BR>
887         * System.out.println("aSampler - PCR6 -- " + NbSamples + " particles");<BR>
888         *     System.out.println("Conflict Z = "+Z+" %");<BR>
889         *     System.out.println(aSampler.state(printMode));<BR>
890         *     System.out.println();<BR>
891         *     //<BR>
892         * <font color="#FF0000" style="font-family: georgia">
893         * aSampler - PCR6 -- 1000000 particles<BR>
894         * Conflict Z = 0.0 %<BR>
895         * bbaTreeA:<BR>
896         * x0000000000000001 -> 0.25761<BR>
897         * x0000000000000002 -> 0.238275<BR>
898         * x0000000000000003 -> 0.105398<BR>
899         * x0000000000000004 -> 0.179867<BR>
900         * x0000000000000005 -> 0.081813<BR>
901         * x0000000000000006 -> 0.056784<BR>
902         * x0000000000000007 -> 0.080253<BR>
```

```
903         * </font><BR>
904         *
905         * <font color="#0000FF" style="font-family: georgia">
906         * Initialize the fuser of <i>aSampler</i> with bba enties <i>aFuser1</i> and
907         * <i>aFuser2</i>, and with the referee function <i>referee4</i> (PCR#);
908         * </font><BR>
909         * aSampler.setFuser(aSampler1, aSampler2, referee5);<BR>
910         * <font color="#0000FF" style="font-family: georgia">
911         * Clear the samples container <i>Samples</i>;
912         * </font><BR>
913         * Samples.clear();<BR>
914         * <font color="#0000FF" style="font-family: georgia">
915         * Build the samples (<i>NbSamples</i> times) fused by means of the fuser and
916         * store them within the container;
917         * </font><BR>
918         * {@code for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());}<↩
                BR>
919         * <font color="#0000FF" style="font-family: georgia">
920         * Learn the bba <i>aSampler</i> from the samples <i>Samples</i> and store the
921         * percentage of conflict samples within <i>Z</i>;
922         * </font><BR>
923         * Z=aSampler.learnFrom(Samples.toArray());<BR>
924         * <font color="#0000FF" style="font-family: georgia">
925         * Print the conflict and the state of <i>aSampler</i> (these results are ↩
                random).
926         * With option <i>printMode=1</i>, the print is done according to the ↩
                proposition
927         * order;
928         * </font><BR>
929         * System.out.println("aSampler - PCR# -- " + NbSamples + " particles");<BR>
930         *     System.out.println("Conflict Z = "+Z+" %");<BR>
931         *     System.out.println(aSampler.state(printMode));<BR>
932         *     System.out.println();<BR>
933         * <font color="#FF0000" style="font-family: georgia">
934         * aSampler - PCR# -- 1000000 particles<BR>
935         * Conflict Z = 0.0 %<BR>
936         * bbaTreeA:<BR>
937         * x0000000000000001 -> 0.257286<BR>
938         * x0000000000000002 -> 0.239343<BR>
939         * x0000000000000003 -> 0.105115<BR>
940         * x0000000000000004 -> 0.179943<BR>
941         * x0000000000000005 -> 0.08186<BR>
942         * x0000000000000006 -> 0.056558<BR>
943         * x0000000000000007 -> 0.079895
944         * </font><BR>
945         *
946         * </code>
947         *
948         */
949        public void RefereeSampler_Comparison() {
950            finalPowerset A=new finalPowerset();
951            A.size(3); A.atomic(0);
952            finalPowerset B=A.instanceNsize(); B.atomic(1);
953            finalPowerset C=A.instanceNsize(); C.atomic(2);
954            //
955            finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);
956            finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);
957            finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);
958            //
959            finalPowerset zero=A.instanceNsize(); zero.zero();
960            finalPowerset one=A.instanceNsize(); one.one();
961            //
962            //
963            RFDempster_Powerset referee1 = new RFDempster_Powerset();
964            RFDisjunctive_Powerset referee2 = new RFDisjunctive_Powerset();
965            RFDuboisPrade_Powerset referee3 = new RFDuboisPrade_Powerset();
966            RFPCR6_Powerset referee4 = new RFPCR6_Powerset();
967            RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();
968            //
969            //
970            finalRefereeSampler_Powerset aSampler1 = new finalRefereeSampler_Powerset↩
                   ();
971            aSampler1.add(A,0.09);
972            aSampler1.add(B,0.2);
```

```
973         aSampler1.add(C,0.02);
974         aSampler1.add(AUB,0.05);
975         aSampler1.add(BUC,0.03);
976         aSampler1.add(CUA,0.1);
977         aSampler1.add(A,0.11);
978         aSampler1.add(one,0.4);
979         //
980         finalRefereeSampler_Powerset aSampler2 = aSampler1.instance();
981         aSampler2.add(A,0.1);
982         aSampler2.add(B,0.1);
983         aSampler2.add(C,0.2);
984         aSampler2.add(AUB,0.2);
985         aSampler2.add(BUC,0.1);
986         aSampler2.add(CUA,0.1);
987         aSampler2.add(one,0.2);
988         //
989         finalRefereeSampler_Powerset aSampler = aSampler1.instance();
990 //
991
992         int printMode=1;
993
994         System.out.println(
995                 "////////////////////////////\n" +
996                 "// Sampling Method\n" +
997                 "////////////////////\n");
998
999         int n;
1000        int NbSamples=1000000;
1001        double Z;
1002        finalRefereeSampler_Powerset Samples= aSampler1.instance();
1003
1004        System.out.println("aSampler1");
1005            System.out.println(aSampler1.state(printMode));
1006            System.out.println();
1007        System.out.println("aSampler2");
1008            System.out.println(aSampler2.state(printMode));
1009            System.out.println();
1010        //
1011        aSampler.setFuser(aSampler1, aSampler2, referee1);
1012        Samples.clear();
1013        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
1014        Z=aSampler.learnFrom(Samples.toArray());
1015        System.out.println("aSampler - Dempster -- " + NbSamples + " particles");
1016            System.out.println("Conflict Z = "+Z+" %");
1017            System.out.println(aSampler.state(printMode));
1018            System.out.println();
1019            //
1020        aSampler.setFuser(aSampler1, aSampler2, referee2);
1021        Samples.clear();
1022        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
1023        Z=aSampler.learnFrom(Samples.toArray());
1024        System.out.println("aSampler - Disjunctive -- " + NbSamples + " particles"↩
                );
1025            System.out.println("Conflict Z = "+Z+" %");
1026            System.out.println(aSampler.state(printMode));
1027            System.out.println();
1028            //
1029        aSampler.setFuser(aSampler1, aSampler2, referee3);
1030        Samples.clear();
1031        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
1032        Z=aSampler.learnFrom(Samples.toArray());
1033        System.out.println("aSampler - Dubois & Prade -- " + NbSamples + " ↩
                particles");
1034            System.out.println("Conflict Z = "+Z+" %");
1035            System.out.println(aSampler.state(printMode));
1036            System.out.println();
1037            //
1038        aSampler.setFuser(aSampler1, aSampler2, referee4);
1039        Samples.clear();
1040        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
1041        Z=aSampler.learnFrom(Samples.toArray());
1042        System.out.println("aSampler - PCR6 -- " + NbSamples + " particles");
1043            System.out.println("Conflict Z = "+Z+" %");
1044            System.out.println(aSampler.state(printMode));
```

```
1045            System.out.println();
1046            //
1047        aSampler.setFuser(aSampler1, aSampler2, referee5);
1048        Samples.clear();
1049        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
1050        Z=aSampler.learnFrom(Samples.toArray());
1051        System.out.println("aSampler - PCR# -- " + NbSamples + " particles");
1052            System.out.println("Conflict Z = "+Z+" %");
1053            System.out.println(aSampler.state(printMode));
1054            System.out.println();
1055
1056    }
1057
1058    /**
1059     * Compare different referee functions applied with RefereeFuser/↩
          RefereeSampler and Powerset.
1060     *
1061     */
1062    public void Referee_Comparison() {
1063        RefereeFuserRTS_Comparison();
1064        RefereeSampler_Comparison();
1065    }
1066
1067    /**
1068     * Test some logical manipulations ({@link Freeboolean}, {@link Superpowerset↩
          },
1069     * {@link Openhyperpowerset}, {@link Closedhyperpowerset}, {@link Powerset}).
1070     *
1071     */
1072    public void logical_tests() {
1073        int theSizeOfFrame = 4;
1074        int theSizeOfFramePowerset = 8;
1075
1076        logical_tests_Freeboolean(theSizeOfFrame);
1077        System.out.println("\n-----------------------------------------\n" +
1078                "-----------------------------------------\n");
1079        logical_tests_Superpowerset(theSizeOfFrame);
1080        System.out.println("\n-----------------------------------------\n" +
1081                "-----------------------------------------\n");
1082        logical_tests_Openhyperpowerset(theSizeOfFrame);
1083        System.out.println("\n-----------------------------------------\n" +
1084                "-----------------------------------------\n");
1085        logical_tests_Closedhyperpowerset(theSizeOfFrame);
1086        System.out.println("\n-----------------------------------------\n" +
1087                "-----------------------------------------\n");
1088        logical_tests_Powerset(theSizeOfFramePowerset);
1089    }
1090
1091    /**
1092     * Test some logical manipulations on Freeboolean.
1093     *
1094     * <BR><BR>
1095     * <b>Detailed code description.</b><BR>
1096     * The following typonomic conventions are used:<BR>
1097     * <code>
1098     * <font color="#0000FF" style="font-family: georgia">
1099     * Blue color is used for commenting the following code.
1100     * </font><BR>
1101     * code formating is used for printing the code.<BR>
1102     * <font color="#FF0000" style="font-family: georgia">
1103     * Red color is used for printing the possible output resulting of the
1104     * previous code.
1105     * </font>
1106     * </code><BR><BR>
1107     * For the output, it is assumed that <i>theSizeOfFrame=4</i> (parameter of ↩
          the method).
1108     * <BR><BR>
1109     * Commented code:
1110     * <BR><BR>
1111     * <code>
1112     * int i;<BR>
1113     * <font color="#0000FF" style="font-family: georgia">
1114     * Declaration of a free Boolean algebra, <i>fFb</i>;
1115     * </font><BR>
```

```
1116      * finalFreeboolean fFb = new finalFreeboolean();<BR>
1117      * <font color="#0000FF" style="font-family: georgia">
1118      * Resize <i>fFb</i> to <i>theSizeOfFrame</i>, the parameter of the method.
1119      * If the resizing failed, produce an error message and exit;
1120      * </font><BR>
1121      * if(fFb.size(theSizeOfFrame)!=theSizeOfFrame) {<BR>
1122      *     System.err.println("Error :: RefereeToolbox_Tutorial.↩
              logical_tests_Freeboolean(int)\n" +<BR>
1123      *         "Exceed authorized bounds!");<BR>
1124      *     System.exit(0);<BR>
1125      * }<BR>
1126      * <font color="#0000FF" style="font-family: georgia">
1127      * Print the title of the method;
1128      * </font><BR>
1129      * System.out.println("\n\n////// Test: Freeboolean - size = " + ↩
              theSizeOfFrame +<BR>
1130      *         "    //\n\n");<BR>
1131      * <font color="#FF0000" style="font-family: georgia">
1132      * ////// Test: Freeboolean - size = 4     //<BR>
1133      * </font>
1134      * <font color="#0000FF" style="font-family: georgia">
1135      * Create the table of atomic propositions, <i>atomF</i>; <BR>
1136      * Create each atomic proposition by instancing and resizing from <i>fFb</i>, ↩
              using
1137      * {@code fFb.instanceNsize();};<BR>
1138      * Define each atomic by applying the method {@link GeneratedLattice#atomic(↩
              int) }; <BR>
1139      * </font>
1140      * finalFreeboolean[] atomF = new finalFreeboolean[theSizeOfFrame];<BR>
1141      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1142      *     atomF[i]=fFb.instanceNsize();<BR>
1143      *     atomF[i].atomic(i);<BR>
1144      * }<BR>
1145      * <font color="#0000FF" style="font-family: georgia">
1146      * Print <i>zero</i>, <i>one</i> and the table of atomic propositions
1147      * (printed with hexadecimal coding);
1148      * </font><BR>
1149      * fFb.zero();<BR>
1150      * System.out.println("zero = "+fFb.state());<BR>
1151      * fFb.one();<BR>
1152      * System.out.println("one = "+fFb.state());<BR>
1153      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1154      *     System.out.println("atomF["+i+"] = " +atomF[i].state());<BR>
1155      * }<BR>
1156      * System.out.println("\n-----------------------------------------\n");<BR>
1157      * <font color="#FF0000" style="font-family: georgia">
1158      * zero = x0000000000000000<BR>
1159      * one = x000000000000ffff<BR>
1160      * atomF[0] = x000000000000aaaa<BR>
1161      * atomF[1] = x000000000000cccc<BR>
1162      * atomF[2] = x000000000000f0f0<BR>
1163      * atomF[3] = x000000000000ff00<BR>
1164      * <BR>
1165      * -----------------------------------------<BR>
1166      * </font>
1167      * <font color="#0000FF" style="font-family: georgia">
1168      * Print the complement of <i>zero</i>, of <i>one</i> and the table of
1169      * complements of the atomic propositions. The complements
1170      * are computed by means of code {@code fFb.complement(atomF[i]); } and
1171      * stored within <i>fFb</i>;
1172      * </font><BR>
1173      * fFb.zero().complement();<BR>
1174      * System.out.println("complement(zero) = "+fFb.state());<BR>
1175      * fFb.one().complement();<BR>
1176      * System.out.println("complement(one) = "+fFb.state());<BR>
1177      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1178      *     fFb.complement(atomF[i]);<BR>
1179      *     System.out.println("complement(atomF["+i+"]) = " +fFb.state()↩
              );<BR>
1180      * }<BR>
1181      * System.out.println("\n-----------------------------------------\n");<BR>
1182      * <font color="#FF0000" style="font-family: georgia">
1183      * complement(zero) = x000000000000ffff<BR>
1184      * complement(one) = x0000000000000000<BR>
```

```
1185      * complement(atomF[0]) = x0000000000005555<BR>
1186      * complement(atomF[1]) = x0000000000003333<BR>
1187      * complement(atomF[2]) = x000000000000f0f<BR>
1188      * complement(atomF[3]) = x00000000000000ff<BR>
1189      * <BR>
1190      * ----------------------------------------<BR>
1191      * </font>
1192      * <font color="#0000FF" style="font-family: georgia">
1193      * Print the cocomplement of <i>zero</i>, of <i>one</i> and the table of
1194      * cocomplements of the atomic propositions. The cocomplements
1195      * are computed by means of code {@code fFb.cocomplement(atomF[i]); } and
1196      * stored within <i>fFb</i>;
1197      * </font><BR>
1198      * fFb.zero().cocomplement();<BR>
1199      * System.out.println("cocomplement(zero) = "+fFb.state());<BR>
1200      * fFb.one().cocomplement();<BR>
1201      * System.out.println("cocomplement(one) = "+fFb.state());<BR>
1202      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1203      *     fFb.cocomplement(atomF[i]);<BR>
1204      *     System.out.println("cocomplement(atomF["+i+"]) = " +fFb.state↩
                ());<BR>
1205      * }<BR>
1206      * System.out.println("\n-----------------------------------------\n");<BR>
1207      * <font color="#FF0000" style="font-family: georgia">
1208      * cocomplement(zero) = x000000000000ffff<BR>
1209      * cocomplement(one) = x0000000000000000<BR>
1210      * cocomplement(atomF[0]) = x0000000000005555<BR>
1211      * cocomplement(atomF[1]) = x0000000000003333<BR>
1212      * cocomplement(atomF[2]) = x000000000000f0f<BR>
1213      * cocomplement(atomF[3]) = x00000000000000ff<BR>
1214      * <BR>
1215      * ----------------------------------------<BR>
1216      * </font>
1217      *
1218      * <font color="#0000FF" style="font-family: georgia">
1219      * The following codes (21 lines) tests the property <i>(A OR B) AND (B OR C) ↩
                = B OR (A AND C )</i>,
1220      * and print the results of the test.
1221      * Propositions <i>A, B, C</i> are defined respectively as the <i>(↩
                theSizeOfFrame/4)</i>-th,
1222      *  <i>(theSizeOfFrame/2)</i>-th, <i>(3*theSizeOfFrame/4)</i>-th atomics ↩
                propositions.
1223      * </font><BR>
1224      * System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C ↩
                )");<BR>
1225      * <font color="#FF0000" style="font-family: georgia">
1226      * ////// Test: (A OR B) AND (B OR C) = B OR (A AND C )<BR>
1227      * </font>
1228      * finalFreeboolean A = atomF[theSizeOfFrame/4].clone();
1229      * <font color="#0000FF" style="font-family: georgia">
1230      * // Method {@link Lattice#clone() } create an exact copy of the proposition↩
                </font><BR>
1231      * finalFreeboolean B = atomF[theSizeOfFrame/2].clone();<BR>
1232      * finalFreeboolean C = atomF[(3*theSizeOfFrame)/4].clone();<BR>
1233      * finalFreeboolean AUB = fFb.instanceNsize();<BR>
1234      * finalFreeboolean BUC = fFb.instanceNsize();<BR>
1235      * finalFreeboolean ANC = fFb.instanceNsize();<BR>
1236      * finalFreeboolean left = fFb.instanceNsize();<BR>
1237      * finalFreeboolean right = fFb.instanceNsize();<BR>
1238      * <font color="#0000FF" style="font-family: georgia">
1239      * Compute <i>A OR B</i> and store it within <i>AUB</i>;<BR>
1240      * Compute <i>B OR C</i> and store it within <i>BUC</i>;<BR>
1241      * </font>
1242      * AUB.or(A, B);<BR>
1243      * BUC.or(B, C);<BR>
1244      * <font color="#0000FF" style="font-family: georgia">
1245      * Compute <i>A AND C</i> and store it within <i>ANC</i>;<BR>
1246      * Compute <i>AUB AND BUC</i> and store it within <i>left</i>;<BR>
1247      * Compute <i>B OR ANC</i> and store it within <i>right</i>;<BR>
1248      * </font>
1249      * ANC.and(A, C);<BR>
1250      * left.and(AUB,BUC);<BR>
1251      * right.or(B,ANC);<BR>
1252      * <font color="#0000FF" style="font-family: georgia">
```

```
1253        * Test if <i>left</i> and <i>right</i> are equal, and store the result within
1254        * <i>isEqual</i>;<BR>
1255        * </font>
1256        * boolean isEqual=left.compareTo(right)==0;<BR>
1257        * <font color="#0000FF" style="font-family: georgia">
1258        * Print propositions <i>A,B,C, left ,right</i> and the boolean <i>isEqual</i>;<↩
                BR>
1259        * </font>
1260        * System.out.println("A = atomF["+theSizeOfFrame/4+"] = " + A.state());<BR>
1261        * System.out.println("B = atomF["+theSizeOfFrame/2+"] = " + B.state());<BR>
1262        * System.out.println("C = atomF["+(3*theSizeOfFrame)/4+"] = " + C.state());<↩
                BR>
1263        * System.out.println("(A OR B) AND (B OR C)  = " + left.state());<BR>
1264        * System.out.println("B OR (A AND C ) = " + right.state());<BR>
1265        * System.out.println("isEqual = "+isEqual);<BR>
1266        * <font color="#FF0000" style="font-family: georgia">
1267        * A = atomF[1] = x000000000000cccc<BR>
1268        * B = atomF[2] = x000000000000f0f0<BR>
1269        * C = atomF[3] = x000000000000ff00<BR>
1270        * (A OR B) AND (B OR C)  = x000000000000fcf0<BR>
1271        * B OR (A AND C ) = x000000000000fcf0<BR>
1272        * isEqual = true<BR>
1273        * </font>
1274        * </code>
1275        *
1276        */
1277       public void logical_tests_Freeboolean(int theSizeOfFrame) {
1278
1279           int i;
1280           finalFreeboolean fFb = new finalFreeboolean();
1281           if(fFb.size(theSizeOfFrame)!=theSizeOfFrame) {
1282               System.err.println("Error :: RefereeToolbox_Tutorial.↩
                       logical_tests_Freeboolean(int)\n" +
1283                       "Exceed authorized bounds!");
1284               System.exit(0);
1285           }
1286           System.out.println("\n\n////// Test: Freeboolean - size = " + ↩
                   theSizeOfFrame +
1287                   "    //\n\n");
1288           finalFreeboolean[] atomF = new finalFreeboolean[theSizeOfFrame];
1289           for(i=0;i<theSizeOfFrame;i++) {
1290               atomF[i]=fFb.instanceNsize();
1291               atomF[i].atomic(i);
1292           }
1293           fFb.zero();
1294           System.out.println("zero = "+fFb.state());
1295           fFb.one();
1296           System.out.println("one = "+fFb.state());
1297           for(i=0;i<theSizeOfFrame;i++) {
1298               System.out.println("atomF["+i+"] = " +atomF[i].state());
1299           }
1300           System.out.println("\n-----------------------------------------\n");
1301           fFb.zero().complement();
1302           System.out.println("complement(zero) = "+fFb.state());
1303           fFb.one().complement();
1304           System.out.println("complement(one) = "+fFb.state());
1305           for(i=0;i<theSizeOfFrame;i++) {
1306               fFb.complement(atomF[i]);
1307               System.out.println("complement(atomF["+i+"]) = " +fFb.state());
1308           }
1309           System.out.println("\n-----------------------------------------\n");
1310           fFb.zero().cocomplement();
1311           System.out.println("cocomplement(zero) = "+fFb.state());
1312           fFb.one().cocomplement();
1313           System.out.println("cocomplement(one) = "+fFb.state());
1314           for(i=0;i<theSizeOfFrame;i++) {
1315               fFb.cocomplement(atomF[i]);
1316               System.out.println("cocomplement(atomF["+i+"]) = " +fFb.state());
1317           }
1318           System.out.println("\n-----------------------------------------\n");
1319
1320           System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C↩
                   )");
1321           finalFreeboolean A = atomF[theSizeOfFrame/4].clone();
```

```
1322            finalFreeboolean B = atomF[theSizeOfFrame/2].clone();
1323            finalFreeboolean C = atomF[(3*theSizeOfFrame)/4].clone();
1324            finalFreeboolean AUB = fFb.instanceNsize();
1325            finalFreeboolean BUC = fFb.instanceNsize();
1326            finalFreeboolean ANC = fFb.instanceNsize();
1327            finalFreeboolean left = fFb.instanceNsize();
1328            finalFreeboolean right = fFb.instanceNsize();
1329            AUB.or(A, B);
1330            BUC.or(B, C);
1331            ANC.and(A, C);
1332            left.and(AUB,BUC);
1333            right.or(B,ANC);
1334            boolean isEqual=left.compareTo(right)==0;
1335            System.out.println("A = atomF["+theSizeOfFrame/4+"] = " + A.state());
1336            System.out.println("B = atomF["+theSizeOfFrame/2+"] = " + B.state());
1337            System.out.println("C = atomF["+(3*theSizeOfFrame)/4+"] = " + C.state());
1338            System.out.println("(A OR B) AND (B OR C)  = " + left.state());
1339            System.out.println("B OR (A AND C ) = " + right.state());
1340            System.out.println("isEqual = "+isEqual);
1341
1342        }
1343
1344        /**
1345         * Test some logical manipulations on Superpowerset.
1346         *
1347         * <BR><BR>
1348         * <b>Detailed code description.</b><BR>
1349         * The following typonomic conventions are used:<BR>
1350         * <code>
1351         * <font color="#0000FF" style="font-family: georgia">
1352         * Blue color is used for commenting the following code.
1353         * </font><BR>
1354         * code formating is used for printing the code.<BR>
1355         * <font color="#FF0000" style="font-family: georgia">
1356         * Red color is used for printing the possible output resulting of the
1357         * previous code.
1358         * </font>
1359         * </code><BR><BR>
1360         * For the output, it is assumed that <i>theSizeOfFrame=4</i> (parameter of ↩
                the method).
1361         * <BR><BR>
1362         * Commented code:
1363         * <BR><BR>
1364         * <code>
1365         * int i;<BR>
1366         * <font color="#0000FF" style="font-family: georgia">
1367         * Declaration of a superpowerset, <i>fSps</i>;
1368         * </font><BR>
1369         * finalSuperpowerset fSps = new finalSuperpowerset();<BR>
1370         * <font color="#0000FF" style="font-family: georgia">
1371         * Resize <i>fSps</i> to <i>theSizeOfFrame</i>, the parameter of the method.
1372         * If the resizing failed, produce an error message and exit;
1373         * </font><BR>
1374         * if(fSps.size(theSizeOfFrame)!=theSizeOfFrame) {<BR>
1375         *     System.err.println("Error :: RefereeToolbox_Tutorial.↩
                logical_tests_Superpowerset(int)\n" +<BR>
1376         *         "Exceed authorized bounds!");<BR>
1377         *     System.exit(0);<BR>
1378         * }<BR>
1379         * <font color="#0000FF" style="font-family: georgia">
1380         * Print the title of the method;
1381         * </font><BR>
1382         * System.out.println("\n\n////// Test: Superpowerset - size = " + ↩
                theSizeOfFrame +<BR>
1383         *         "    //\n\n");<BR>
1384         * <font color="#FF0000" style="font-family: georgia">
1385         * ////// Test: Superpowerset - size = 4     //<BR>
1386         * </font>
1387         * <font color="#0000FF" style="font-family: georgia">
1388         * Create the table of atomic propositions, <i>atomS</i>; <BR>
1389         * Create each atomic proposition by instancing and resizing from <i>fSps</i>,↩
                using
1390         * {@code fSps.instanceNsize();};<BR>
1391         * Define each atomic by applying the method {@link GeneratedLattice#atomic(↩
```

```
     int) }; <BR>
1392  * </font>
1393  * finalSuperpowerset[] atomS = new finalSuperpowerset[theSizeOfFrame];<BR>
1394  * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1395  *     atomS[i]=fSps.instanceNsize();<BR>
1396  *     atomS[i].atomic(i);<BR>
1397  * }<BR>
1398  * <font color="#0000FF" style="font-family: georgia">
1399  * Print <i>zero</i>, <i>one</i> and the table of atomic propositions (printed
1400  * with hexadecimal coding);
1401  * </font><BR>
1402  * fSps.zero();<BR>
1403  * System.out.println("zero = "+fSps.state());<BR>
1404  * fSps.one();<BR>
1405  * System.out.println("one = "+fSps.state());<BR>
1406  * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1407  *     System.out.println("atomS["+i+"] = " +atomS[i].state());<BR>
1408  * }<BR>
1409  * System.out.println("\n-------------------------------------\n");<BR>
1410  * <font color="#FF0000" style="font-family: georgia">
1411  * zero = x0000000000000000<BR>
1412  * one = x000000000000fffe<BR>
1413  * atomS[0] = x000000000000aaaa<BR>
1414  * atomS[1] = x000000000000cccc<BR>
1415  * atomS[2] = x000000000000f0f0<BR>
1416  * atomS[3] = x000000000000ff00<BR>
1417  * <BR>
1418  * ---------------------------------------<BR>
1419  * </font>
1420  * <font color="#0000FF" style="font-family: georgia">
1421  * Print the complement of <i>zero</i>, of <i>one</i> and the table of
1422  * complements of the atomic propositions. The complements
1423  * are computed by means of code {@code fSps.complement(atom[i]); } and
1424  * stored within <i>fSps</i>;
1425  * </font><BR>
1426  * fSps.zero().complement();<BR>
1427  * System.out.println("complement(zero) = "+fSps.state());<BR>
1428  * fSps.one().complement();<BR>
1429  * System.out.println("complement(one) = "+fSps.state());<BR>
1430  * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1431  *     fSps.complement(atomS[i]);<BR>
1432  *     System.out.println("complement(atomS["+i+"]) = " +fSps.state↩
     ());<BR>
1433  * }<BR>
1434  * System.out.println("\n-------------------------------------\n");<BR>
1435  * <font color="#FF0000" style="font-family: georgia">
1436  * complement(zero) = x000000000000fffe<BR>
1437  * complement(one) = x0000000000000000<BR>
1438  * complement(atomS[0]) = x0000000000005554<BR>
1439  * complement(atomS[1]) = x0000000000003332<BR>
1440  * complement(atomS[2]) = x0000000000000f0e<BR>
1441  * complement(atomS[3]) = x00000000000000fe<BR>
1442  * <BR>
1443  * ---------------------------------------<BR>
1444  * </font>
1445  * <font color="#0000FF" style="font-family: georgia">
1446  * Print the cocomplement of <i>zero</i>, of <i>one</i> and the table of
1447  * cocomplements of the atomic propositions. The cocomplements
1448  * are computed by means of code {@code fSps.cocomplement(atomS[i]); } and
1449  * stored within <i>fSps</i>;
1450  * </font><BR>
1451  * fSps.zero().cocomplement();<BR>
1452  * System.out.println("cocomplement(zero) = "+fSps.state());<BR>
1453  * fSps.one().cocomplement();<BR>
1454  * System.out.println("cocomplement(one) = "+fSps.state());<BR>
1455  * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1456  *     fSps.cocomplement(atomS[i]);<BR>
1457  *     System.out.println("cocomplement(atomS["+i+"]) = " +fSps.↩
     state());<BR>
1458  * }<BR>
1459  *
1460  * System.out.println("\n-------------------------------------\n");<BR>
1461  * <font color="#FF0000" style="font-family: georgia">
1462  * cocomplement(zero) = x000000000000fffe<BR>
```

```
1463      * cocomplement(one) = x0000000000000000<BR>
1464      * cocomplement(atomS[0]) = x0000000000005554<BR>
1465      * cocomplement(atomS[1]) = x0000000000003332<BR>
1466      * cocomplement(atomS[2]) = x000000000000f0e<BR>
1467      * cocomplement(atomS[3]) = x00000000000000fe<BR>
1468      * <BR>
1469      * ----------------------------------------<BR>
1470      * </font>
1471      *
1472      * <font color="#0000FF" style="font-family: georgia">
1473      * The following codes (21 lines) tests the property <i>(A OR B) AND (B OR C) ↩
              = B OR (A AND C )</i>,
1474      * and print the results of the test.
1475      * Propositions <i>A, B, C</i> are defined respectively as the <i>(↩
              theSizeOfFrame/4)</i>-th,
1476      *  <i>(theSizeOfFrame/2)</i>-th, <i>(3*theSizeOfFrame/4)</i>-th atomics ↩
              propositions.
1477      * </font><BR>
1478      * System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C ↩
              )");<BR>
1479      * <font color="#FF0000" style="font-family: georgia">
1480      * ////// Test: (A OR B) AND (B OR C) = B OR (A AND C )<BR>
1481      * </font>
1482      * finalSuperpowerset A = atomS[theSizeOfFrame/4].clone();
1483      * <font color="#0000FF" style="font-family: georgia">
1484      * // Method {@link Lattice#clone() } create an exact copy of the proposition↩
              </font><BR>
1485      * finalSuperpowerset B = atomS[theSizeOfFrame/2].clone();<BR>
1486      * finalSuperpowerset C = atomS[(3*theSizeOfFrame)/4].clone();<BR>
1487      * finalSuperpowerset AUB = fSps.instanceNsize();<BR>
1488      * finalSuperpowerset BUC = fSps.instanceNsize();<BR>
1489      * finalSuperpowerset ANC = fSps.instanceNsize();<BR>
1490      * finalSuperpowerset left = fSps.instanceNsize();<BR>
1491      * finalSuperpowerset right = fSps.instanceNsize();<BR>
1492      * <font color="#0000FF" style="font-family: georgia">
1493      * Compute <i>A OR B</i> and store it within <i>AUB</i>;<BR>
1494      * Compute <i>B OR C</i> and store it within <i>BUC</i>;<BR>
1495      * </font>
1496      * AUB.or(A, B);<BR>
1497      * BUC.or(B, C);<BR>
1498      * <font color="#0000FF" style="font-family: georgia">
1499      * Compute <i>A AND C</i> and store it within <i>ANC</i>;<BR>
1500      * Compute <i>AUB AND BUC</i> and store it within <i>left</i>;<BR>
1501      * Compute <i>B OR ANC</i> and store it within <i>right</i>;<BR>
1502      * </font>
1503      * ANC.and(A, C);<BR>
1504      * left.and(AUB,BUC);<BR>
1505      * right.or(B,ANC);<BR>
1506      * <font color="#0000FF" style="font-family: georgia">
1507      * Test if <i>left</i> and <i>right</i> are equal, and store the result within
1508      * <i>isEqual</i>;<BR>
1509      * </font>
1510      * boolean isEqual=left.compareTo(right)==0;<BR>
1511      * <font color="#0000FF" style="font-family: georgia">
1512      * Print propositions <i>A,B,C,left,right</i> and the boolean <i>isEqual</i>;<↩
              BR>
1513      * </font>
1514      * System.out.println("A = atomS["+theSizeOfFrame/4+"] = " + A.state());<BR>
1515      * System.out.println("B = atomS["+theSizeOfFrame/2+"] = " + B.state());<BR>
1516      * System.out.println("C = atomS["+(3*theSizeOfFrame)/4+"] = " + C.state());<↩
              BR>
1517      * System.out.println("(A OR B) AND (B OR C)  = " + left.state());<BR>
1518      * System.out.println("B OR (A AND C ) = " + right.state());<BR>
1519      * System.out.println("isEqual = "+isEqual);<BR>
1520      * <font color="#FF0000" style="font-family: georgia">
1521      * A = atomS[1] = x0000000000000cccc<BR>
1522      * B = atomS[2] = x000000000000f0f0<BR>
1523      * C = atomS[3] = x000000000000ff00<BR>
1524      * (A OR B) AND (B OR C)  = x0000000000000fcf0<BR>
1525      * B OR (A AND C ) = x000000000000fcf0<BR>
1526      * isEqual = true<BR>
1527      * </font>
1528      *
1529      */
```

```
1530    public void logical_tests_Superpowerset(int theSizeOfFrame) {
1531
1532        int i;
1533        finalSuperpowerset fSps = new finalSuperpowerset();
1534        if(fSps.size(theSizeOfFrame)!=theSizeOfFrame) {
1535            System.err.println("Error :: RefereeToolbox_Tutorial.↩
                    logical_tests_Superpowerset(int)\n" +
1536                    "Exceed authorized bounds!");
1537            System.exit(0);
1538        }
1539        System.out.println("\n\n////// Test: Superpowerset - size = " + ↩
                theSizeOfFrame +
1540                "    //\n\n");
1541        finalSuperpowerset[] atomS = new finalSuperpowerset[theSizeOfFrame];
1542        for(i=0;i<theSizeOfFrame;i++) {
1543            atomS[i]=fSps.instanceNsize();
1544            atomS[i].atomic(i);
1545        }
1546        fSps.zero();
1547        System.out.println("zero = "+fSps.state());
1548        fSps.one();
1549        System.out.println("one = "+fSps.state());
1550        for(i=0;i<theSizeOfFrame;i++) {
1551            System.out.println("atomS["+i+"] = " +atomS[i].state());
1552        }
1553        System.out.println("\n-----------------------------------------\n");
1554        fSps.zero().complement();
1555        System.out.println("complement(zero) = "+fSps.state());
1556        fSps.one().complement();
1557        System.out.println("complement(one) = "+fSps.state());
1558        for(i=0;i<theSizeOfFrame;i++) {
1559            fSps.complement(atomS[i]);
1560            System.out.println("complement(atomS["+i+"]) = " +fSps.state());
1561        }
1562        System.out.println("\n-----------------------------------------\n");
1563        fSps.zero().cocomplement();
1564        System.out.println("cocomplement(zero) = "+fSps.state());
1565        fSps.one().cocomplement();
1566        System.out.println("cocomplement(one) = "+fSps.state());
1567        for(i=0;i<theSizeOfFrame;i++) {
1568            fSps.cocomplement(atomS[i]);
1569            System.out.println("cocomplement(atomS["+i+"]) = " +fSps.state());
1570        }
1571
1572        System.out.println("\n-----------------------------------------\n");
1573
1574        System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C↩
              )");
1575        finalSuperpowerset A = atomS[theSizeOfFrame/4].clone();
1576        finalSuperpowerset B = atomS[theSizeOfFrame/2].clone();
1577        finalSuperpowerset C = atomS[(3*theSizeOfFrame)/4].clone();
1578        finalSuperpowerset AUB = fSps.instanceNsize();
1579        finalSuperpowerset BUC = fSps.instanceNsize();
1580        finalSuperpowerset ANC = fSps.instanceNsize();
1581        finalSuperpowerset left = fSps.instanceNsize();
1582        finalSuperpowerset right = fSps.instanceNsize();
1583        AUB.or(A, B);
1584        BUC.or(B, C);
1585        ANC.and(A, C);
1586        left.and(AUB,BUC);
1587        right.or(B,ANC);
1588        boolean isEqual=left.compareTo(right)==0;
1589        System.out.println("A = atomS["+theSizeOfFrame/4+"] = " + A.state());
1590        System.out.println("B = atomS["+theSizeOfFrame/2+"] = " + B.state());
1591        System.out.println("C = atomS["+(3*theSizeOfFrame)/4+"] = " + C.state());
1592        System.out.println("(A OR B) AND (B OR C)  = " + left.state());
1593        System.out.println("B OR (A AND C ) = " + right.state());
1594        System.out.println("isEqual = "+isEqual);
1595
1596    }
1597
1598    /**
1599     * Test some logical manipulations on Openhyperpowerset.
1600     *
```

```
1601      * <BR><BR>
1602      * <b>Detailed code description.</b><BR>
1603      * The following typonomic conventions are used:<BR>
1604      * <code>
1605      * <font color="#0000FF" style="font-family: georgia">
1606      * Blue color is used for commenting the following code.
1607      * </font><BR>
1608      * code formating is used for printing the code.<BR>
1609      * <font color="#FF0000" style="font-family: georgia">
1610      * Red color is used for printing the possible output resulting of the
1611      * previous code.
1612      * </font>
1613      * </code><BR><BR>
1614      * For the output, it is assumed that <i>theSizeOfFrame=4</i> (parameter of ↩
              the method).
1615      * <BR><BR>
1616      * Commented code:
1617      * <BR><BR>
1618      * <code>
1619      * int i;<BR>
1620      *
1621      * <font color="#0000FF" style="font-family: georgia">
1622      * Declaration of an open hyperpowerset, <i>fOhps</i>;
1623      * </font><BR>
1624      * finalOpenhyperpowerset fOhps = new finalOpenhyperpowerset();<BR>
1625      * <font color="#0000FF" style="font-family: georgia">
1626      * Resize <i>fOhps</i> to <i>theSizeOfFrame</i>, the parameter of the method.
1627      * If the resizing failed, produce an error message and exit;
1628      * </font><BR>
1629      * if(fOhps.size(theSizeOfFrame)!=theSizeOfFrame) {<BR>
1630      *     System.err.println("Error :: RefereeToolbox_Tutorial.↩
              logical_tests_Openhyperpowerset(int)\n" +<BR>
1631      *         "Exceed authorized bounds!");<BR>
1632      *     System.exit(0);<BR>
1633      * }<BR>
1634      * <font color="#0000FF" style="font-family: georgia">
1635      * Print the title of the method;
1636      * </font><BR>
1637      * System.out.println("\n\n////// Test: Openhyperpowerset - size = " + ↩
              theSizeOfFrame +<BR>
1638      *         "    //\n\n");<BR>
1639      * <font color="#FF0000" style="font-family: georgia">
1640      * ////// Test: Openhyperpowerset - size = 4     //<BR>
1641      * </font>
1642      * <font color="#0000FF" style="font-family: georgia">
1643      * Create the table of atomic propositions, <i>atom</i>; <BR>
1644      * Create each atomic proposition by instancing and resizing from <i>fOhps</i↩
              >, using
1645      * {@code fOhps.instanceNsize();};<BR>
1646      * Define each atomic by applying the method {@link GeneratedLattice#atomic(↩
              int) }; <BR>
1647      * </font>
1648      * finalOpenhyperpowerset[] atom = new finalOpenhyperpowerset[theSizeOfFrame↩
              ];<BR>
1649      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1650      *     atom[i]=fOhps.instanceNsize();<BR>
1651      *     atom[i].atomic(i);<BR>
1652      * }<BR>
1653      * <font color="#0000FF" style="font-family: georgia">
1654      * Print <i>zero</i>, <i>one</i> and the table of atomic propositions (printed
1655      * with hexadecimal coding);
1656      * </font><BR>
1657      * fOhps.zero();<BR>
1658      * System.out.println("zero = "+fOhps.state());<BR>
1659      * fOhps.one();<BR>
1660      * System.out.println("one = "+fOhps.state());<BR>
1661      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1662      *     System.out.println("atom["+i+"] = " +atom[i].state());<BR>
1663      * }<BR>
1664      * System.out.println("\n-----------------------------------------\n");<BR>
1665      * <font color="#FF0000" style="font-family: georgia">
1666      * zero = x0000000000000000<BR>
1667      * one = x000000000000ffff<BR>
1668      * atom[0] = x000000000000aaaa<BR>
```

```
1669      * atom[1] = x000000000000cccc<BR>
1670      * atom[2] = x000000000000f0f0<BR>
1671      * atom[3] = x000000000000ff00<BR>
1672      * <BR>
1673      * ---------------------------------------<BR>
1674      * </font>
1675      * <font color="#0000FF" style="font-family: georgia">
1676      * Print the complement of <i>zero</i>, of <i>one</i> and the table of
1677      * complements of the atomic propositions. The complements
1678      * are computed by means of code {@code fOhps.complement(atom[i]); } and
1679      * stored within <i>fOhps</i>;
1680      * </font><BR>
1681      * fOhps.zero().complement();<BR>
1682      * System.out.println("complement(zero) = "+fOhps.state());<BR>
1683      * fOhps.one().complement();<BR>
1684      * System.out.println("complement(one) = "+fOhps.state());<BR>
1685      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1686      *     fOhps.complement(atom[i]);<BR>
1687      *     System.out.println("complement(atom["+i+"]) = " +fOhps.state↩
                ());<BR>
1688      * }<BR>
1689      *
1690      * System.out.println("\n-------------------------------------\n");<BR>
1691      * <font color="#FF0000" style="font-family: georgia">
1692      * complement(zero) = x000000000000ffff<BR>
1693      * complement(one) = x0000000000000000<BR>
1694      * complement(atom[0]) = x0000000000000000<BR>
1695      * complement(atom[1]) = x0000000000000000<BR>
1696      * complement(atom[2]) = x0000000000000000<BR>
1697      * complement(atom[3]) = x0000000000000000<BR>
1698      * <BR>
1699      * ---------------------------------------<BR>
1700      * </font>
1701      * <font color="#0000FF" style="font-family: georgia">
1702      * Print the cocomplement of <i>zero</i>, of <i>one</i> and the table of
1703      * cocomplements of the atomic propositions. The cocomplements
1704      * are computed by means of code {@code fOhps.cocomplement(atom[i]); } and
1705      * stored within <i>fOhps</i>;
1706      * </font><BR>
1707      * fOhps.zero().cocomplement();<BR>
1708      * System.out.println("cocomplement(zero) = "+fOhps.state());<BR>
1709      * fOhps.one().cocomplement();<BR>
1710      * System.out.println("cocomplement(one) = "+fOhps.state());<BR>
1711      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1712      *     fOhps.cocomplement(atom[i]);<BR>
1713      *     System.out.println("cocomplement(atom["+i+"]) = " +fOhps.↩
                state());<BR>
1714      * }<BR>
1715      * System.out.println("\n-------------------------------------\n");<BR>
1716      * <font color="#FF0000" style="font-family: georgia">
1717      * cocomplement(zero) = x000000000000ffff<BR>
1718      * cocomplement(one) = x0000000000000000<BR>
1719      * cocomplement(atom[0]) = x000000000000ffff<BR>
1720      * cocomplement(atom[1]) = x000000000000ffff<BR>
1721      * cocomplement(atom[2]) = x000000000000ffff<BR>
1722      * cocomplement(atom[3]) = x000000000000ffff<BR>
1723      * <BR>
1724      * ---------------------------------------<BR>
1725      * </font>
1726      *
1727      * <font color="#0000FF" style="font-family: georgia">
1728      * The following codes (21 lines) tests the property <i>(A OR B) AND (B OR C) ↩
                = B OR (A AND C )</i>,
1729      * and print the results of the test.
1730      * Propositions <i>A, B, C</i> are defined respectively as the <i>(↩
                theSizeOfFrame/4)</i>-th,
1731      *  <i>(theSizeOfFrame/2)</i>-th, <i>(3*theSizeOfFrame/4)</i>-th atomics ↩
                propositions.
1732      * </font><BR>
1733      * System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C ↩
                )");<BR>
1734      * <font color="#FF0000" style="font-family: georgia">
1735      * ////// Test: (A OR B) AND (B OR C) = B OR (A AND C )<BR>
1736      * </font>
```

```
1737      * finalOpenhyperpowerset A = atom[theSizeOfFrame/4].clone();
1738      * <font color="#0000FF" style="font-family: georgia">
1739      * // Method {@link Lattice#clone() } create an exact copy of the proposition↩
            </font><BR>
1740      * finalOpenhyperpowerset B = atom[theSizeOfFrame/2].clone();<BR>
1741      * finalOpenhyperpowerset C = atom[(3*theSizeOfFrame)/4].clone();<BR>
1742      * finalOpenhyperpowerset AUB = fOhps.instanceNsize();<BR>
1743      * finalOpenhyperpowerset BUC = fOhps.instanceNsize();<BR>
1744      * finalOpenhyperpowerset ANC = fOhps.instanceNsize();<BR>
1745      * finalOpenhyperpowerset left = fOhps.instanceNsize();<BR>
1746      * finalOpenhyperpowerset right = fOhps.instanceNsize();<BR>
1747      * <font color="#0000FF" style="font-family: georgia">
1748      * Compute <i>A OR B</i> and store it within <i>AUB</i>;<BR>
1749      * Compute <i>B OR C</i> and store it within <i>BUC</i>;<BR>
1750      * </font>
1751      * AUB.or(A, B);<BR>
1752      * BUC.or(B, C);<BR>
1753      * <font color="#0000FF" style="font-family: georgia">
1754      * Compute <i>A AND C</i> and store it within <i>ANC</i>;<BR>
1755      * Compute <i>AUB AND BUC</i> and store it within <i>left</i>;<BR>
1756      * Compute <i>B OR ANC</i> and store it within <i>right</i>;<BR>
1757      * </font>
1758      * ANC.and(A, C);<BR>
1759      * left.and(AUB,BUC);<BR>
1760      * right.or(B,ANC);<BR>
1761      * <font color="#0000FF" style="font-family: georgia">
1762      * Test if <i>left</i> and <i>right</i> are equal, and store the result within
1763      * <i>isEqual</i>;<BR>
1764      * </font>
1765      * boolean isEqual=left.compareTo(right)==0;<BR>
1766      * <font color="#0000FF" style="font-family: georgia">
1767      * Print propositions <i>A,B,C,left,right</i> and the boolean <i>isEqual</i>;<↩
            BR>
1768      * </font>
1769      * System.out.println("A = atom["+theSizeOfFrame/4+"] = " + A.state());<BR>
1770      * System.out.println("B = atom["+theSizeOfFrame/2+"] = " + B.state());<BR>
1771      * System.out.println("C = atom["+(3*theSizeOfFrame)/4+"] = " + C.state());<BR↩
            >
1772      * System.out.println("(A OR B) AND (B OR C)  = " + left.state());<BR>
1773      * System.out.println("B OR (A AND C ) = " + right.state());<BR>
1774      * System.out.println("isEqual = "+isEqual);<BR>
1775      * <font color="#FF0000" style="font-family: georgia">
1776      * A = atom[1] = x000000000000cccc<BR>
1777      * B = atom[2] = x000000000000f0f0<BR>
1778      * C = atom[3] = x000000000000ff00<BR>
1779      * (A OR B) AND (B OR C)  = x000000000000fcf0<BR>
1780      * B OR (A AND C ) = x000000000000fcf0<BR>
1781      * isEqual = true<BR>
1782      * </font>
1783      * </code>
1784      *
1785      */
1786     public void logical_tests_Openhyperpowerset(int theSizeOfFrame) {
1787
1788         int i;
1789         finalOpenhyperpowerset fOhps = new finalOpenhyperpowerset();
1790         if(fOhps.size(theSizeOfFrame)!=theSizeOfFrame) {
1791             System.err.println("Error :: RefereeToolbox_Tutorial.↩
                    logical_tests_Openhyperpowerset(int)\n" +
1792                     "Exceed authorized bounds!");
1793             System.exit(0);
1794         }
1795         System.out.println("\n\n////// Test: Openhyperpowerset - size = " + ↩
                theSizeOfFrame +
1796                 "    //\n\n");
1797         finalOpenhyperpowerset[] atom = new finalOpenhyperpowerset[theSizeOfFrame↩
                ];
1798         for(i=0;i<theSizeOfFrame;i++) {
1799             atom[i]=fOhps.instanceNsize();
1800             atom[i].atomic(i);
1801         }
1802         fOhps.zero();
1803         System.out.println("zero = "+fOhps.state());
1804         fOhps.one();
```

```
1805            System.out.println("one = "+fOhps.state());
1806            for(i=0;i<theSizeOfFrame;i++) {
1807                System.out.println("atom["+i+"] = " +atom[i].state());
1808            }
1809            System.out.println("\n-----------------------------------------\n");
1810            fOhps.zero().complement();
1811            System.out.println("complement(zero) = "+fOhps.state());
1812            fOhps.one().complement();
1813            System.out.println("complement(one) = "+fOhps.state());
1814            for(i=0;i<theSizeOfFrame;i++) {
1815                fOhps.complement(atom[i]);
1816                System.out.println("complement(atom["+i+"]) = " +fOhps.state());
1817            }
1818            System.out.println("\n-----------------------------------------\n");
1819            fOhps.zero().cocomplement();
1820            System.out.println("cocomplement(zero) = "+fOhps.state());
1821            fOhps.one().cocomplement();
1822            System.out.println("cocomplement(one) = "+fOhps.state());
1823            for(i=0;i<theSizeOfFrame;i++) {
1824                fOhps.cocomplement(atom[i]);
1825                System.out.println("cocomplement(atom["+i+"]) = " +fOhps.state());
1826            }
1827            System.out.println("\n-----------------------------------------\n");
1828
1829            System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C↩
                    )");
1830            finalOpenhyperpowerset A = atom[theSizeOfFrame/4].clone();
1831            finalOpenhyperpowerset B = atom[theSizeOfFrame/2].clone();
1832            finalOpenhyperpowerset C = atom[(3*theSizeOfFrame)/4].clone();
1833            finalOpenhyperpowerset AUB = fOhps.instanceNsize();
1834            finalOpenhyperpowerset BUC = fOhps.instanceNsize();
1835            finalOpenhyperpowerset ANC = fOhps.instanceNsize();
1836            finalOpenhyperpowerset left = fOhps.instanceNsize();
1837            finalOpenhyperpowerset right = fOhps.instanceNsize();
1838            AUB.or(A, B);
1839            BUC.or(B, C);
1840            ANC.and(A, C);
1841            left.and(AUB,BUC);
1842            right.or(B,ANC);
1843            boolean isEqual=left.compareTo(right)==0;
1844            System.out.println("A = atom["+theSizeOfFrame/4+"] = " + A.state());
1845            System.out.println("B = atom["+theSizeOfFrame/2+"] = " + B.state());
1846            System.out.println("C = atom["+(3*theSizeOfFrame)/4+"] = " + C.state());
1847            System.out.println("(A OR B) AND (B OR C)  = " + left.state());
1848            System.out.println("B OR (A AND C ) = " + right.state());
1849            System.out.println("isEqual = "+isEqual);
1850
1851        }
1852
1853        /**
1854         * Test some logical manipulations on Closedhyperpowerset.
1855         *
1856         * <BR><BR>
1857         * <b>Detailed code description.</b><BR>
1858         * The following typonomic conventions are used:<BR>
1859         * <code>
1860         * <font color="#0000FF" style="font-family: georgia">
1861         * Blue color is used for commenting the following code.
1862         * </font><BR>
1863         * code formating is used for printing the code.<BR>
1864         * <font color="#FF0000" style="font-family: georgia">
1865         * Red color is used for printing the possible output resulting of the
1866         * previous code.
1867         * </font>
1868         * </code><BR><BR>
1869         * For the output, it is assumed that <i>theSizeOfFrame=4</i> (parameter of ↩
                the method).
1870         * <BR><BR>
1871         * Commented code:
1872         * <BR><BR>
1873         * <code>
1874         * int i;<BR>
1875         * <font color="#0000FF" style="font-family: georgia">
1876         * Declaration of a closed hyperpowerset, <i>fChps</i>;
```

```
1877      * </font><BR>
1878      * finalClosedhyperpowerset fChps = new finalClosedhyperpowerset();<BR>
1879      * <font color="#0000FF" style="font-family: georgia">
1880      * Resize <i>fChps</i> to <i>theSizeOfFrame</i>, the parameter of the method.
1881      * If the resizing failed, produce an error message and exit;
1882      * </font><BR>
1883      * if(fChps.size(theSizeOfFrame)!=theSizeOfFrame) {<BR>
1884      *     System.err.println("Error :: RefereeToolbox_Tutorial.↩
            logical_tests_Closedhyperpowerset(int)\n" +<BR>
1885      *         "Exceed authorized bounds!");<BR>
1886      *     System.exit(0);<BR>
1887      * }<BR>
1888      * <font color="#0000FF" style="font-family: georgia">
1889      * Print the title of the method;
1890      * </font><BR>
1891      * System.out.println("\n\n////// Test: Closedhyperpowerset - size = " + ↩
            theSizeOfFrame +<BR>
1892      *         "    //\n\n");<BR>
1893      * <font color="#FF0000" style="font-family: georgia">
1894      * ////// Test: Closedhyperpowerset - size = 4     //<BR>
1895      * </font>
1896      * <font color="#0000FF" style="font-family: georgia">
1897      * Create the table of atomic propositions, <i>atomC</i>; <BR>
1898      * Create each atomic proposition by instancing and resizing from <i>fChps</i↩
            >, using
1899      * {@code fChps.instanceNsize();};<BR>
1900      * Define each atomic by applying the method {@link GeneratedLattice#atomic(↩
            int) }; <BR>
1901      * </font>
1902      * finalClosedhyperpowerset[] atomC = new finalClosedhyperpowerset[↩
            theSizeOfFrame];<BR>
1903      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1904      *     atomC[i]=fChps.instanceNsize();<BR>
1905      *     atomC[i].atomic(i);<BR>
1906      * }<BR>
1907      * <font color="#0000FF" style="font-family: georgia">
1908      * Print <i>zero</i>, <i>one</i> and the table of atomic propositions (printed
1909      * with hexadecimal coding);
1910      * </font><BR>
1911      * fChps.zero();<BR>
1912      * System.out.println("zero = "+fChps.state());<BR>
1913      * fChps.one();<BR>
1914      * System.out.println("one = "+fChps.state());<BR>
1915      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1916      *     System.out.println("atomC["+i+"] = " +atomC[i].state());<BR>
1917      * }<BR>
1918      * System.out.println("\n-------------------------------------\n");<BR>
1919      * <font color="#FF0000" style="font-family: georgia">
1920      * zero = x0000000000000000 <BR>
1921      * one = x000000000000fffe <BR>
1922      * atomC[0] = x000000000000aaaa <BR>
1923      * atomC[1] = x000000000000cccc <BR>
1924      * atomC[2] = x000000000000f0f0 <BR>
1925      * atomC[3] = x000000000000ff00 <BR>
1926      * <BR>
1927      * ---------------------------------------<BR>
1928      * </font>
1929      *
1930      * <font color="#0000FF" style="font-family: georgia">
1931      * Print the complement of <i>zero</i>, of <i>one</i> and the table of ↩
            complements
1932      * of the atomic propositions. The complements are computed by means of code
1933      * {@code fChps.complement(atomC[i]); } and stored within <i>fChps</i>;
1934      * </font><BR>
1935      * fChps.zero().complement();<BR>
1936      * System.out.println("complement(zero) = "+fChps.state());<BR>
1937      * fChps.one().complement();<BR>
1938      * System.out.println("complement(one) = "+fChps.state());<BR>
1939      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1940      *     fChps.complement(atomC[i]);<BR>
1941      *     System.out.println("complement(atomC["+i+"]) = " +fChps.state↩
            ());<BR>
1942      * }<BR>
1943      * System.out.println("\n-------------------------------------\n");<BR>
```

```
1944      * <font color="#FF0000" style="font-family: georgia">
1945      * complement(zero) = x000000000000fffe<BR>
1946      * complement(one) = x0000000000000000<BR>
1947      * complement(atomC[0]) = x0000000000000000<BR>
1948      * complement(atomC[1]) = x0000000000000000<BR>
1949      * complement(atomC[2]) = x0000000000000000<BR>
1950      * complement(atomC[3]) = x0000000000000000<BR>
1951      * <BR>
1952      * ----------------------------------------<BR>
1953      * </font>
1954      *
1955      * <font color="#0000FF" style="font-family: georgia">
1956      * Print the cocomplement of <i>zero</i>, of <i>one</i> and the table of cocomplements
1957      * of the atomic propositions. The cocomplements
1958      * are computed by means of code {@code fChps.cocomplement(atomC[i]); } and
1959      * stored within <i>fChps</i>;
1960      * </font><BR>
1961      * fChps.zero().cocomplement();<BR>
1962      * System.out.println("cocomplement(zero) = "+fChps.state());<BR>
1963      * fChps.one().cocomplement();<BR>
1964      * System.out.println("cocomplement(one) = "+fChps.state());<BR>
1965      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
1966      *     fChps.cocomplement(atomC[i]);<BR>
1967      *     System.out.println("cocomplement(atomC["+i+"]) = " +fChps.state());<BR>
1968      * }<BR>
1969      * System.out.println("\n----------------------------------------\n");<BR>
1970      * <font color="#FF0000" style="font-family: georgia">
1971      * cocomplement(zero) = x000000000000fffe<BR>
1972      * cocomplement(one) = x0000000000000000<BR>
1973      * cocomplement(atomC[0]) = x000000000000fffc<BR>
1974      * cocomplement(atomC[1]) = x000000000000fffa<BR>
1975      * cocomplement(atomC[2]) = x000000000000ffee<BR>
1976      * cocomplement(atomC[3]) = x000000000000fefe<BR>
1977      * <BR>
1978      * ----------------------------------------<BR>
1979      * </font>
1980      *
1981      * <font color="#0000FF" style="font-family: georgia">
1982      * The following codes (21 lines) tests the property <i>(A OR B) AND (B OR C) = B OR (A AND C )</i>,
1983      * and print the results of the test.
1984      * Propositions <i>A, B, C</i> are defined respectively as the <i>(theSizeOfFrame/4)</i>-th,
1985      *  <i>(theSizeOfFrame/2)</i>-th, <i>(3*theSizeOfFrame/4)</i>-th atomics propositions.
1986      * </font><BR>
1987      * System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C )");<BR>
1988      * <font color="#FF0000" style="font-family: georgia">
1989      * ////// Test: (A OR B) AND (B OR C) = B OR (A AND C )<BR>
1990      * </font>
1991      * finalClosedhyperpowerset A = atomC[theSizeOfFrame/4].clone();
1992      * <font color="#0000FF" style="font-family: georgia">
1993      * // Method {@link Lattice#clone() } create an exact copy of the proposition</font><BR>
1994      * finalClosedhyperpowerset B = atomC[theSizeOfFrame/2].clone();<BR>
1995      * finalClosedhyperpowerset C = atomC[(3*theSizeOfFrame)/4].clone();<BR>
1996      * finalClosedhyperpowerset AUB = fChps.instanceNsize();<BR>
1997      * finalClosedhyperpowerset BUC = fChps.instanceNsize();<BR>
1998      * finalClosedhyperpowerset ANC = fChps.instanceNsize();<BR>
1999      * finalClosedhyperpowerset left = fChps.instanceNsize();<BR>
2000      * finalClosedhyperpowerset right = fChps.instanceNsize();<BR>
2001      * <font color="#0000FF" style="font-family: georgia">
2002      * Compute <i>A OR B</i> and store it within <i>AUB</i>;<BR>
2003      * Compute <i>B OR C</i> and store it within <i>BUC</i>;<BR>
2004      * </font>
2005      * AUB.or(A, B);<BR>
2006      * BUC.or(B, C);<BR>
2007      * <font color="#0000FF" style="font-family: georgia">
2008      * Compute <i>A AND C</i> and store it within <i>ANC</i>;<BR>
2009      * Compute <i>AUB AND BUC</i> and store it within <i>left</i>;<BR>
2010      * Compute <i>B OR ANC</i> and store it within <i>right</i>;<BR>
```

```
2011      * </font>
2012      * ANC.and(A, C);<BR>
2013      * left.and(AUB,BUC);<BR>
2014      * right.or(B,ANC);<BR>
2015      * <font color="#0000FF" style="font-family: georgia">
2016      * Test if <i>left</i> and <i>right</i> are equal, and store the result within
2017      * <i>isEqual</i>;<BR>
2018      * </font>
2019      * boolean isEqual=left.compareTo(right)==0;<BR>
2020      * <font color="#0000FF" style="font-family: georgia">
2021      * Print propositions <i>A,B,C,left,right</i> and the boolean <i>isEqual</i>;<↩
              BR>
2022      * </font>
2023      * System.out.println("A = atomC["+theSizeOfFrame/4+"] = " + A.state());<BR>
2024      * System.out.println("B = atomC["+theSizeOfFrame/2+"] = " + B.state());<BR>
2025      * System.out.println("C = atomC["+(3*theSizeOfFrame)/4+"] = " + C.state());<↩
              BR>
2026      * System.out.println("(A OR B) AND (B OR C)  = " + left.state());<BR>
2027      * System.out.println("B OR (A AND C ) = " + right.state());<BR>
2028      * System.out.println("isEqual = "+isEqual);<BR>
2029      * <font color="#FF0000" style="font-family: georgia">
2030      * A = atomC[1] = x0000000000000cccc<BR>
2031      * B = atomC[2] = x000000000000f0f0<BR>
2032      * C = atomC[3] = x000000000000ff00<BR>
2033      * (A OR B) AND (B OR C)  = x000000000000fcf0<BR>
2034      * B OR (A AND C ) = x000000000000fcf0<BR>
2035      * isEqual = true<BR>
2036      * </font>
2037      * </code>
2038      *
2039      */
2040     public void logical_tests_Closedhyperpowerset(int theSizeOfFrame) {
2041
2042         int i;
2043         finalClosedhyperpowerset fChps = new finalClosedhyperpowerset();
2044         if(fChps.size(theSizeOfFrame)!=theSizeOfFrame) {
2045             System.err.println("Error :: RefereeToolbox_Tutorial.↩
                      logical_tests_Closedhyperpowerset(int)\n" +
2046                     "Exceed authorized bounds!");
2047             System.exit(0);
2048         }
2049         System.out.println("\n\n////// Test: Closedhyperpowerset - size = " + ↩
              theSizeOfFrame +
2050                     "    //\n\n");
2051         finalClosedhyperpowerset[] atomC = new finalClosedhyperpowerset[↩
              theSizeOfFrame];
2052         for(i=0;i<theSizeOfFrame;i++) {
2053             atomC[i]=fChps.instanceNsize();
2054             atomC[i].atomic(i);
2055         }
2056         fChps.zero();
2057         System.out.println("zero = "+fChps.state());
2058         fChps.one();
2059         System.out.println("one = "+fChps.state());
2060         for(i=0;i<theSizeOfFrame;i++) {
2061             System.out.println("atomC["+i+"] = " +atomC[i].state());
2062         }
2063         System.out.println("\n-----------------------------------------\n");
2064         fChps.zero().complement();
2065         System.out.println("complement(zero) = "+fChps.state());
2066         fChps.one().complement();
2067         System.out.println("complement(one) = "+fChps.state());
2068         for(i=0;i<theSizeOfFrame;i++) {
2069             fChps.complement(atomC[i]);
2070             System.out.println("complement(atomC["+i+"]) = " +fChps.state());
2071         }
2072         System.out.println("\n-----------------------------------------\n");
2073         fChps.zero().cocomplement();
2074         System.out.println("cocomplement(zero) = "+fChps.state());
2075         fChps.one().cocomplement();
2076         System.out.println("cocomplement(one) = "+fChps.state());
2077         for(i=0;i<theSizeOfFrame;i++) {
2078             fChps.cocomplement(atomC[i]);
2079             System.out.println("cocomplement(atomC["+i+"]) = " +fChps.state());
```

```
2080            }
2081
2082            System.out.println("\n-------------------------------------------\n");
2083
2084            System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C↩
                    )");
2085            finalClosedhyperpowerset A = atomC[theSizeOfFrame/4].clone();
2086            finalClosedhyperpowerset B = atomC[theSizeOfFrame/2].clone();
2087            finalClosedhyperpowerset C = atomC[(3*theSizeOfFrame)/4].clone();
2088            finalClosedhyperpowerset AUB = fChps.instanceNsize();
2089            finalClosedhyperpowerset BUC = fChps.instanceNsize();
2090            finalClosedhyperpowerset ANC = fChps.instanceNsize();
2091            finalClosedhyperpowerset left = fChps.instanceNsize();
2092            finalClosedhyperpowerset right = fChps.instanceNsize();
2093            AUB.or(A, B);
2094            BUC.or(B, C);
2095            ANC.and(A, C);
2096            left.and(AUB,BUC);
2097            right.or(B,ANC);
2098            boolean isEqual=left.compareTo(right)==0;
2099            System.out.println("A = atomC["+theSizeOfFrame/4+"] = " + A.state());
2100            System.out.println("B = atomC["+theSizeOfFrame/2+"] = " + B.state());
2101            System.out.println("C = atomC["+(3*theSizeOfFrame)/4+"] = " + C.state());
2102            System.out.println("(A OR B) AND (B OR C)  = " + left.state());
2103            System.out.println("B OR (A AND C ) = " + right.state());
2104            System.out.println("isEqual = "+isEqual);
2105
2106        }
2107
2108        /**
2109         * Test some logical manipulations on Powerset.
2110         *
2111         * <BR><BR>
2112         * <b>Detailed code description.</b><BR>
2113         * The following typonomic conventions are used:<BR>
2114         * <code>
2115         * <font color="#0000FF" style="font-family: georgia">
2116         * Blue color is used for commenting the following code.
2117         * </font><BR>
2118         * code formating is used for printing the code.<BR>
2119         * <font color="#FF0000" style="font-family: georgia">
2120         * Red color is used for printing the possible output resulting of the
2121         * previous code.
2122         * </font>
2123         * </code><BR><BR>
2124         * For the output, it is assumed that <i>theSizeOfFrame=8</i> (parameter of ↩
                the method).
2125         * <BR><BR>
2126         * Commented code:
2127         * <BR><BR>
2128         * <code>
2129         * int i;<BR>
2130         * <font color="#0000FF" style="font-family: georgia">
2131         * Declaration of a powerset, <i>fPset</i>;
2132         * </font><BR>
2133         * finalPowerset fPset = new finalPowerset();<BR>
2134         * <font color="#0000FF" style="font-family: georgia">
2135         * Resize <i>fPset</i> to <i>theSizeOfFrame</i>, the parameter of the method.
2136         * If the resizing failed, produce an error message and exit;
2137         * </font><BR>
2138         * if(fPset.size(theSizeOfFrame)!=theSizeOfFrame) {<BR>
2139         *     System.err.println("Error :: RefereeToolbox_Tutorial.↩
                logical_tests_Closedhyperpowerset(int)\n" +<BR>
2140         *         "Exceed authorized bounds!");<BR>
2141         *     System.exit(0);<BR>
2142         * }<BR>
2143         * //<BR>
2144         * <font color="#0000FF" style="font-family: georgia">
2145         * Print the title of the method;
2146         * </font><BR>
2147         * System.out.println("\n\n////// Test: Powerset - size = " + theSizeOfFrame ↩
                +<BR>
2148         *         "    //\n\n");<BR>
2149         * <font color="#FF0000" style="font-family: georgia">
```

```
2150      * ////// Test: Powerset - size = 8     //<BR>
2151      * </font>
2152      * <font color="#0000FF" style="font-family: georgia">
2153      * Create the table of atomic propositions, <i>atomPset</i>; <BR>
2154      * Create each atomic proposition by instancing and resizing from <i>fPset</i>, using
2155      * {@code fPset.instanceNsize();};<BR>
2156      * Define each atomic by applying the method {@link GeneratedLattice#atomic(int) }; <BR>
2157      * </font>
2158      * finalPowerset[] atomPset = new finalPowerset[theSizeOfFrame];<BR>
2159      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
2160      *     atomPset[i]=fPset.instanceNsize();<BR>
2161      *     atomPset[i].atomic(i);<BR>
2162      * }<BR>
2163      * <font color="#0000FF" style="font-family: georgia">
2164      * Print the table of atomic propositions (printed with hexadecimal coding);
2165      * </font><BR>
2166      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
2167      *     System.out.println("atomPset["+i+"]" +<BR>
2168      *         " = " +atomPset[i].state());<BR>
2169      * }<BR>
2170      * System.out.println("\n-------------------------------------\n");<BR>
2171      * <font color="#FF0000" style="font-family: georgia">
2172      * atomPset[0] = x0000000000000001<BR>
2173      * atomPset[1] = x0000000000000002<BR>
2174      * atomPset[2] = x0000000000000004<BR>
2175      * atomPset[3] = x0000000000000008<BR>
2176      * atomPset[4] = x0000000000000010<BR>
2177      * atomPset[5] = x0000000000000020<BR>
2178      * atomPset[6] = x0000000000000040<BR>
2179      * atomPset[7] = x0000000000000080<BR>
2180      * <BR>
2181      * ---------------------------------------<BR>
2182      * </font>
2183      * <font color="#0000FF" style="font-family: georgia">
2184      * Print the table of complements of the atomic propositions. The complements
2185      * are computed by means of code {@code fPset.complement(atomPset[i]); } and
2186      * stored within <i>fPset</i>;
2187      * </font><BR>
2188      * {@code for(i=0;i<theSizeOfFrame;i++) } {<BR>
2189      *     fPset.complement(atomPset[i]);<BR>
2190      *     System.out.println("complement(atomPset["+i+"])" +<BR>
2191      *         " = " +fPset.state());<BR>
2192      * }<BR>
2193      * System.out.println("\n-------------------------------------\n");<BR>
2194      * <font color="#FF0000" style="font-family: georgia">
2195      * complement(atomPset[0]) = x00000000000000fe<BR>
2196      * complement(atomPset[1]) = x00000000000000fd<BR>
2197      * complement(atomPset[2]) = x00000000000000fb<BR>
2198      * complement(atomPset[3]) = x00000000000000f7<BR>
2199      * complement(atomPset[4]) = x00000000000000ef<BR>
2200      * complement(atomPset[5]) = x00000000000000df<BR>
2201      * complement(atomPset[6]) = x00000000000000bf<BR>
2202      * complement(atomPset[7]) = x000000000000007f<BR>
2203      * <BR>
2204      * ---------------------------------------<BR>
2205      * </font>
2206      * <font color="#0000FF" style="font-family: georgia">
2207      * Print the table of cocomplements of the atomic propositions. The cocomplements
2208      * are computed by means of code {@code fPset.cocomplement(atomPset[i]); } and
2209      * stored within <i>fPset</i>;
2210      * </font><BR>
2211      * {@code for(i=0;i<theSizeOfFrame;i++) }{<BR>
2212      *     fPset.cocomplement(atomPset[i]);<BR>
2213      *     System.out.println("cocomplement(atomPset["+i+"])" +<BR>
2214      *         " = " +fPset.state());<BR>
2215      * }<BR>
2216      *
2217      * System.out.println("\n-------------------------------------\n");<BR>
2218      * <font color="#FF0000" style="font-family: georgia">
2219      * cocomplement(atomPset[0]) = x00000000000000fe<BR>
2220      * cocomplement(atomPset[1]) = x00000000000000fd<BR>
```

```
2221      * cocomplement(atomPset[2]) = x00000000000000fb<BR>
2222      * cocomplement(atomPset[3]) = x00000000000000f7<BR>
2223      * cocomplement(atomPset[4]) = x00000000000000ef<BR>
2224      * cocomplement(atomPset[5]) = x00000000000000df<BR>
2225      * cocomplement(atomPset[6]) = x00000000000000bf<BR>
2226      * cocomplement(atomPset[7]) = x000000000000007f<BR>
2227      * <BR>
2228      * ----------------------------------------<BR>
2229      * </font>
2230      *
2231      * <font color="#0000FF" style="font-family: georgia">
2232      * The following codes (21 lines) tests the property <i>(A OR B) AND (B OR C) ↩
              = B OR (A AND C )</i>,
2233      * and print the results of the test.
2234      * Propositions <i>A, B, C</i> are defined respectively as the <i>(↩
              theSizeOfFrame/4)</i>-th,
2235      *  <i>(theSizeOfFrame/2)</i>-th, <i>(3*theSizeOfFrame/4)</i>-th atomics ↩
              propositions.
2236      * </font><BR>
2237      * System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C ↩
              )");<BR>
2238      * <font color="#FF0000" style="font-family: georgia">
2239      * ////// Test: (A OR B) AND (B OR C) = B OR (A AND C )<BR>
2240      * </font>
2241      * finalPowerset A = atomPset[theSizeOfFrame/4].clone(); <font color="#0000FF"↩
              style="font-family: georgia">
2242      * // Method {@link Lattice#clone() } create an exact copy of the proposition↩
              </font><BR>
2243      * finalPowerset B = atomPset[theSizeOfFrame/2].clone();<BR>
2244      * finalPowerset C = atomPset[(3*theSizeOfFrame)/4].clone();<BR>
2245      * finalPowerset AUB = fPset.instanceNsize();<BR>
2246      * finalPowerset BUC = fPset.instanceNsize();<BR>
2247      * finalPowerset ANC = fPset.instanceNsize();<BR>
2248      * finalPowerset left = fPset.instanceNsize();<BR>
2249      * finalPowerset right = fPset.instanceNsize();<BR>
2250      * <font color="#0000FF" style="font-family: georgia">
2251      * Compute <i>A OR B</i> and store it within <i>AUB</i>;<BR>
2252      * Compute <i>B OR C</i> and store it within <i>BUC</i>;<BR>
2253      * </font>
2254      * AUB.or(A, B);<BR>
2255      * BUC.or(B, C);<BR>
2256      * <font color="#0000FF" style="font-family: georgia">
2257      * Compute <i>A AND C</i> and store it within <i>ANC</i>;<BR>
2258      * Compute <i>AUB AND BUC</i> and store it within <i>left</i>;<BR>
2259      * Compute <i>B OR ANC</i> and store it within <i>right</i>;<BR>
2260      * </font>
2261      * ANC.and(A, C);<BR>
2262      * left.and(AUB,BUC);<BR>
2263      * right.or(B,ANC);<BR>
2264      * <font color="#0000FF" style="font-family: georgia">
2265      * Test if <i>left</i> and <i>right</i> are equal, and store the result within
2266      * <i>isEqual</i>;<BR>
2267      * </font>
2268      * boolean isEqual=left.compareTo(right)==0;<BR>
2269      * <font color="#0000FF" style="font-family: georgia">
2270      * Print propositions <i>A,B,C, left , right</i> and the boolean <i>isEqual</i>;<↩
              BR>
2271      * </font>
2272      * System.out.println("A = atomPset["+theSizeOfFrame/4+"] = " + A.state());<BR↩
              >
2273      * System.out.println("B = atomPset["+theSizeOfFrame/2+"] = " + B.state());<BR↩
              >
2274      * System.out.println("C = atomPset["+(3*theSizeOfFrame)/4+"] = " + C.state())↩
              ;<BR>
2275      * System.out.println("(A OR B) AND (B OR C)  = " + left.state());<BR>
2276      * System.out.println("B OR (A AND C ) = " + right.state());<BR>
2277      * System.out.println("isEqual = "+isEqual);<BR>
2278      * <font color="#FF0000" style="font-family: georgia">
2279      * A = atomPset[2] = x0000000000000004<BR>
2280      * B = atomPset[4] = x0000000000000010<BR>
2281      * C = atomPset[6] = x0000000000000040<BR>
2282      * (A OR B) AND (B OR C)  = x0000000000000010<BR>
2283      * B OR (A AND C ) = x0000000000000010<BR>
2284      * isEqual = true<BR>
```

```
2285        * </font>
2286        *
2287        * </code>
2288        *
2289        */
2290       public void logical_tests_Powerset(int theSizeOfFrame) {
2291
2292           int i;
2293           finalPowerset fPset = new finalPowerset();
2294           if(fPset.size(theSizeOfFrame)!=theSizeOfFrame) {
2295               System.err.println("Error :: RefereeToolbox_Tutorial.↩
                       logical_tests_Closedhyperpowerset(int)\n" +
2296                       "Exceed authorized bounds!");
2297               System.exit(0);
2298           }
2299           //
2300           System.out.println("\n\n////// Test: Powerset - size = " + theSizeOfFrame ↩
                   +
2301                   "    //\n\n");
2302           finalPowerset[] atomPset = new finalPowerset[theSizeOfFrame];
2303           for(i=0;i<theSizeOfFrame;i++) {
2304                   atomPset[i]=fPset.instanceNsize();
2305                   atomPset[i].atomic(i);
2306           }
2307           for(i=0;i<theSizeOfFrame;i++) {
2308               System.out.println("atomPset["+i+"]" +
2309                   " = " +atomPset[i].state());
2310           }
2311           System.out.println("\n-------------------------------------------\n");
2312           for(i=0;i<theSizeOfFrame;i++) {
2313               fPset.complement(atomPset[i]);
2314               System.out.println("complement(atomPset["+i+"])" +
2315                   " = " +fPset.state());
2316           }
2317           System.out.println("\n-------------------------------------------\n");
2318           for(i=0;i<theSizeOfFrame;i++) {
2319               fPset.cocomplement(atomPset[i]);
2320               System.out.println("cocomplement(atomPset["+i+"])" +
2321                   " = " +fPset.state());
2322           }
2323
2324           System.out.println("\n-------------------------------------------\n");
2325
2326           System.out.println("\n\n////// Test: (A OR B) AND (B OR C) = B OR (A AND C↩
                    )");
2327           finalPowerset A = atomPset[theSizeOfFrame/4].clone();
2328           finalPowerset B = atomPset[theSizeOfFrame/2].clone();
2329           finalPowerset C = atomPset[(3*theSizeOfFrame)/4].clone();
2330           finalPowerset AUB = fPset.instanceNsize();
2331           finalPowerset BUC = fPset.instanceNsize();
2332           finalPowerset ANC = fPset.instanceNsize();
2333           finalPowerset left = fPset.instanceNsize();
2334           finalPowerset right = fPset.instanceNsize();
2335           AUB.or(A, B);
2336           BUC.or(B, C);
2337           ANC.and(A, C);
2338           left.and(AUB,BUC);
2339           right.or(B,ANC);
2340           boolean isEqual=left.compareTo(right)==0;
2341           System.out.println("A = atomPset["+theSizeOfFrame/4+"] = " + A.state());
2342           System.out.println("B = atomPset["+theSizeOfFrame/2+"] = " + B.state());
2343           System.out.println("C = atomPset["+(3*theSizeOfFrame)/4+"] = " + C.state()↩
                   );
2344           System.out.println("(A OR B) AND (B OR C)  = " + left.state());
2345           System.out.println("B OR (A AND C ) = " + right.state());
2346           System.out.println("isEqual = "+isEqual);
2347
2348       }
2349
2350       /**
2351        * Compare different referee functions applied to the fusion of 3 bba.
2352        * Both sampling and relaxed approaches are considered.
2353        *
2354        * This example works like {@link RefereeToolbox_Tutorial#↩
```

```
           RefereeFuserRTS_Comparison() }
2355       * and {@link RefereeToolbox_Tutorial#RefereeSampler_Comparison() } so that
2356       * it is not explained in details.
2357       * However, it is noticed that the combinations work here on three sources,
2358       * denoted <i>aFuser1</i>, <i>aFuser2</i>, <i>aFuser3</i> for the relaxed approach,
2359       * and <i>aSampler1</i>, <i>aSampler2</i>, <i>aSampler3</i> for the sampled approach.
2360       * In order to compute on these tree sources, it is necessary to use an <i>ArrayList</i>
2361       * containing the sources, and apply the methods:<BR>
2362       * {@link BBARefereeFuser#fuse(java.util.ArrayList, RefereeToolbox.RefereeFunctionDefault) }
2363       * for the relaxed approach<BR>
2364       * {@link SampledBBARefereeFuser#setFuser(java.util.ArrayList, RefereeToolbox.RefereeFunctionDefault) }
2365       * for the sampling approach<BR>
2366       * <BR>
2367       * Typically, the fusion by means of PCR# is done by the following codes:<BR>
2368       * <code>
2369       * // referee function definition<BR>
2370       * RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();<BR>
2371       * // [...]<BR>
2372       * // Array creation<BR>
2373       * <font color="#FF0000" >
2374       * ArrayList<finalRefereeFuserRTS_Powerset> bbaArrayR =<BR>
2375       *         new {@code ArrayList<finalRefereeFuserRTS_Powerset>}();<BR>
2376       * bbaArrayR.add(aFuser1);<BR>
2377       * bbaArrayR.add(aFuser2);<BR>
2378       * bbaArrayR.add(aFuser3);<BR>
2379       * </font>
2380       * // [...]<BR>
2381       * // Combination<BR>
2382       * <font color="#FF0000" >
2383       * aFuser.fuse(bbaArrayR,referee5);<BR>
2384       * </font>
2385       * </code>
2386       * for the relaxed approach,<BR>
2387       * <code>
2388       * // referee function definition<BR>
2389       * RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();<BR>
2390       * // [...]<BR>
2391       * // Array creation<BR>
2392       * <font color="#FF0000" >
2393       * ArrayList<finalRefereeSampler_Powerset> bbaArrayS =<BR>
2394       *         new {@code ArrayList<finalRefereeSampler_Powerset>}();<BR>
2395       * bbaArrayS.add(aSampler1);<BR>
2396       * bbaArrayS.add(aSampler2);<BR>
2397       * bbaArrayS.add(aSampler3);<BR>
2398       * </font>
2399       * // [...]<BR>
2400       * // Combination<BR>
2401       * <font color="#FF0000" >
2402       * aSampler.setFuser(bbaArrayS, referee5);<BR>
2403       * </font>
2404       * Samples.clear();<BR>
2405       * {@code for(n=0;n<NbSamples;n++) } Samples.add(aSampler.makeFusedSample());<BR>
2406       * Z=aSampler.learnFrom(Samples.toArray());<BR>
2407       * </code>
2408       * for the sampled approach,<BR>
2409       *<BR>
2410       * where <i>aFuser</i> and <i>aSampler</i> are output bba for each
2411       * respective approaches.
2412       *
2413       */
2414      public void Referee_On_3_Entries(){
2415
2416          int printMode=1;
2417          //
2418          finalPowerset A=new finalPowerset();
2419          A.size(3); A.atomic(0);
```

```
2420            finalPowerset B=A.instanceNsize(); B.atomic(1);
2421            finalPowerset C=A.instanceNsize(); C.atomic(2);
2422            //
2423            finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);
2424            finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);
2425            finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);
2426            //
2427            finalPowerset zero=A.instanceNsize(); zero.zero();
2428            finalPowerset one=A.instanceNsize(); one.one();
2429            //
2430            //
2431            RFDempster_Powerset referee1 = new RFDempster_Powerset();
2432            RFDisjunctive_Powerset referee2 = new RFDisjunctive_Powerset();
2433            RFDuboisPrade_Powerset referee3 = new RFDuboisPrade_Powerset();
2434            RFPCR6_Powerset referee4 = new RFPCR6_Powerset();
2435            RFPCRSharp_Powerset referee5  =new RFPCRSharp_Powerset();
2436            //
2437            //
2438            finalRefereeFuserRTS_Powerset aFuser1 = new finalRefereeFuserRTS_Powerset↩
                    ();
2439            aFuser1.add(A,0.6);
2440            aFuser1.add(AUB,0.4);
2441            //
2442            finalRefereeFuserRTS_Powerset aFuser2 = aFuser1.instance();
2443            aFuser2.add(A,0.3);
2444            aFuser2.add(CUA,0.7);
2445            //
2446            finalRefereeFuserRTS_Powerset aFuser3 = aFuser1.instance();
2447            aFuser3.add(B,0.8);
2448            aFuser3.add(one,0.2);
2449            //
2450            ArrayList<finalRefereeFuserRTS_Powerset> bbaArrayR =
2451                    new ArrayList<finalRefereeFuserRTS_Powerset>();
2452            bbaArrayR.add(aFuser1);
2453            bbaArrayR.add(aFuser2);
2454            bbaArrayR.add(aFuser3);
2455            //
2456            finalRefereeFuserRTS_Powerset aFuser = aFuser1.instance();
2457
2458            finalRefereeSampler_Powerset aSampler1 = new finalRefereeSampler_Powerset↩
                    ();
2459            aSampler1.addAll(aFuser1.toArray());
2460            //
2461            finalRefereeSampler_Powerset aSampler2 = aSampler1.instance();
2462            aSampler2.addAll(aFuser2.toArray());
2463            //
2464            finalRefereeSampler_Powerset aSampler3 = aSampler1.instance();
2465            aSampler3.addAll(aFuser3.toArray());
2466            //
2467            ArrayList<finalRefereeSampler_Powerset> bbaArrayS =
2468                    new ArrayList<finalRefereeSampler_Powerset>();
2469            bbaArrayS.add(aSampler1);
2470            bbaArrayS.add(aSampler2);
2471            bbaArrayS.add(aSampler3);
2472            //
2473            finalRefereeSampler_Powerset aSampler = aSampler1.instance();
2474            //
2475            //
2476            System.out.println(
2477                    "////////////////////////////\n" +
2478                    "// Relaxed Method\n" +
2479                    "////////////////////\n");
2480            System.out.println("aFuser1");
2481                System.out.println(aFuser1.state(printMode));
2482                System.out.println();
2483            System.out.println("aFuser2");
2484                System.out.println(aFuser2.state(printMode));
2485                System.out.println();
2486            System.out.println("aFuser3");
2487                System.out.println(aFuser3.state(printMode));
2488                System.out.println();
2489            aFuser.fuse(bbaArrayR,referee1);
2490            System.out.println("aFuser - Dempster");
2491                System.out.println("Conflict Z = "+aFuser.conflict()+" %");
```

```
2492            System.out.println(aFuser.state(printMode));
2493            System.out.println();
2494        aFuser.fuse(bbaArrayR,referee2);
2495        System.out.println("aFuser - Disjunctive");
2496            System.out.println("Conflict Z = "+aFuser.conflict()+" %");
2497            System.out.println(aFuser.state(printMode));
2498            System.out.println();
2499        aFuser.fuse(bbaArrayR,referee3);
2500        System.out.println("aFuser - Dubois & Prade");
2501            System.out.println("Conflict Z = "+aFuser.conflict()+" %");
2502            System.out.println(aFuser.state(printMode));
2503            System.out.println();
2504        aFuser.fuse(bbaArrayR,referee4);
2505        System.out.println("aFuser - PCR6");
2506            System.out.println("Conflict Z = "+aFuser.conflict()+" %");
2507            System.out.println(aFuser.state(printMode));
2508            System.out.println();
2509        aFuser.fuse(bbaArrayR,referee5);
2510        System.out.println("aFuser - PCR#");
2511            System.out.println("Conflict Z = "+aFuser.conflict()+" %");
2512            System.out.println(aFuser.state(printMode));
2513            System.out.println();
2514
2515        System.out.println(
2516                "////////////////////////////\n" +
2517                "// Sampling Method\n" +
2518                "////////////////////\n");
2519
2520        int n;
2521        int NbSamples=1000000;
2522        double Z;
2523        finalRefereeSampler_Powerset Samples= aSampler1.instance();
2524
2525        System.out.println("aSampler1");
2526            System.out.println(aSampler1.state(printMode));
2527            System.out.println();
2528        System.out.println("aSampler2");
2529            System.out.println(aSampler2.state(printMode));
2530            System.out.println();
2531        System.out.println("aSampler3");
2532            System.out.println(aSampler3.state(printMode));
2533            System.out.println();
2534        //
2535        aSampler.setFuser(bbaArrayS, referee1);
2536        Samples.clear();
2537        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
2538        Z=aSampler.learnFrom(Samples.toArray());
2539        System.out.println("aSampler - Dempster -- " + NbSamples + " particles");
2540            System.out.println("Conflict Z = "+Z+" %");
2541            System.out.println(aSampler.state(printMode));
2542            System.out.println();
2543            //
2544        aSampler.setFuser(bbaArrayS, referee2);
2545        Samples.clear();
2546        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
2547        Z=aSampler.learnFrom(Samples.toArray());
2548        System.out.println("aSampler - Disjunctive -- " + NbSamples + " particles"↩
                );
2549            System.out.println("Conflict Z = "+Z+" %");
2550            System.out.println(aSampler.state(printMode));
2551            System.out.println();
2552            //
2553        aSampler.setFuser(bbaArrayS, referee3);
2554        Samples.clear();
2555        for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
2556        Z=aSampler.learnFrom(Samples.toArray());
2557        System.out.println("aSampler - Dubois & Prade -- " + NbSamples + " ↩
                particles");
2558            System.out.println("Conflict Z = "+Z+" %");
2559            System.out.println(aSampler.state(printMode));
2560            System.out.println();
2561            //
2562        aSampler.setFuser(bbaArrayS, referee4);
2563        Samples.clear();
```

```
2564            for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
2565            Z=aSampler.learnFrom(Samples.toArray());
2566            System.out.println("aSampler - PCR6 -- " + NbSamples + " particles");
2567                System.out.println("Conflict Z = "+Z+" %");
2568                System.out.println(aSampler.state(printMode));
2569                System.out.println();
2570                //
2571            aSampler.setFuser(bbaArrayS, referee5);
2572            Samples.clear();
2573            for(n=0;n<NbSamples;n++) Samples.add(aSampler.makeFusedSample());
2574            Z=aSampler.learnFrom(Samples.toArray());
2575            System.out.println("aSampler - PCR# -- " + NbSamples + " particles");
2576                System.out.println("Conflict Z = "+Z+" %");
2577                System.out.println(aSampler.state(printMode));
2578                System.out.println();
2579
2580        }
2581
2582
2583        /**
2584         * Demonstrate the creation of a new Referee Function.
2585         * <BR><BR>
2586         * In this example, a new referee function is created from the referee ↩
                function
2587         * for Dempster-Shafer by excluding a proposition <i>excludedProposition</i>.
2588         * More precisely:<BR>
2589         *     - If <i>excludedProposition=null</i>, just do as for Dempster↩
                -Shafer,
2590         * <BR>
2591         *     - Otherwise:<BR>
2592         *         - Let <i>(X , 1.)</i> be the referee output for↩
                 Dempster-Shafer,<BR>
2593         *         - If <i>excludedProposition CONTAINS X </i>, ↩
                then return <i>(zero , 1.)</i>,
2594         * that is a full rejection,<BR>
2595         *         - Otherwise, return <i>(X AND cocomplement(↩
                excludedProposition)
2596         * , 1.)</i>, that is <i>X</i> truncated by <i>excludedProposition</i>.
2597         * <BR><BR>
2598         * Except for the referee function creation, this example works like
2599         * {@link RefereeToolbox_Tutorial#RefereeFuserRTS_Comparison() }, so that the
2600         * subsequent explained code only concern the Referee Function creation.
2601         *
2602         * <BR><BR>
2603         * <b>Detailed code description.</b><BR>
2604         * The following typonomic conventions are used:<BR>
2605         * <code>
2606         * <font color="#0000FF" style="font-family: georgia">
2607         * Blue color is used for commenting the following code.
2608         * </font><BR>
2609         * code formating is used for printing the code.<BR>
2610         * </code>
2611         * <BR>
2612         * Commented code:
2613         * <BR><BR>
2614         * <code>
2615         * <font color="#0000FF" style="font-family: georgia">
2616         * Creation of a class <i>myRefereeFunction</i> extending the class
2617         * {@link RefereeFunctionDempster } (a referee function for Dempster-Shafer).
2618         * This definition is generic, depending on a class <i>Prop</i> which is a ↩
                complemented lattice,
2619         * that is {@code Prop extends ComplementedLattice<Prop> };
2620         * </font><BR>
2621         * class {@code myRefereeFunction<Prop extends ComplementedLattice<Prop> > }<↩
                BR>
2622         *                    
2623         * {@code extends RefereeFunctionDempster<Prop> }{<BR>
2624         * <font color="#0000FF" style="font-family: georgia">
2625         * Declaration of the excluded proposition <i>excludedProposition</i>. By
2626         * default, it is undefined;
2627         * </font><BR>
2628         *     public Prop excludedProposition = null;<BR>
2629         * <font color="#0000FF" style="font-family: georgia">
2630         * Redefinition (overriding) of method {@code ArrayList<Assignment<Prop>> ↩
```

```
            refereeFunction(...) };
2631        * </font><BR>
2632        *     @Override<BR>
2633        *     public {@code ArrayList<Assignment<Prop>> }refereeFunction(<↩
                BR>
2634        *                      ↩
                ;  
2635        * {@code ArrayList<Assignment<Prop>> assignIn , }<BR>
2636        *                      ↩
                ;  
2637        * {@code ArrayList<minAssignment<Prop>>  bbaIn) }{<BR>
2638        * <font color="#0000FF" style="font-family: georgia">
2639        * In case <i>excludedProposition</i> is undefined, do like the referee ↩
                function
2640        * for Dempster-Shafer;
2641        * </font><BR>
2642        *         if(excludedProposition==null) return
2643        * super.refereeFunction(assignIn , bbaIn);<BR>
2644        * <font color="#0000FF" style="font-family: georgia">
2645        * Otherwise, compute the referee function for Dempster-Shafer and store the
2646        * result within <i>output</i> ;
2647        * </font><BR>
2648        *         {@code ArrayList<Assignment<Prop>> }
2649        * output=super.refereeFunction(assignIn , bbaIn);<BR>
2650        * <font color="#0000FF" style="font-family: georgia">
2651        * If the attribute of <i>output</i> is contained by <i>excludedProposition</i↩
                >
2652        * </font><BR>
2653        *         if(excludedProposition.contains(output.get(0).↩
                attribute))<BR>
2654        * <font color="#0000FF" style="font-family: georgia">
2655        * Then, set <i>output</i> to a rejection;
2656        * </font><BR>
2657        *             output.get(0).attribute.zero();<↩
                BR>
2658        * <font color="#0000FF" style="font-family: georgia">
2659        * Otherwise, set <i>output</i> to its conjunction with <i>cocomplement(↩
                excludedProposition)</i>;
2660        * </font><BR>
2661        *         else output.get(0).attribute.and(output.get(0).↩
                attribute,<BR>
2662        *                      ↩
                ;  
2663        * excludedProposition.clone().cocomplement());<BR>
2664        * <font color="#0000FF" style="font-family: georgia">
2665        * At last, return <i>output</i>;
2666        * </font><BR>
2667        *         return output;<BR>
2668        *     }<BR>
2669        * <BR>
2670        * }<BR>
2671        *<BR>
2672        * <font color="#0000FF" style="font-family: georgia">
2673        * <b>Typical usage:</b>
2674        * <BR><BR>
2675        * Create a referee function <i>myReferee</i> typed <i>myRefereeFunction</i>
2676        * and instantiated for powerset;
2677        * </font><BR>
2678        * {@code myRefereeFunction<finalPowerset> myReferee = }<BR>
2679        *                
2680        * {@code new myRefereeFunction<finalPowerset >(); }<BR>
2681        * <font color="#0000FF" style="font-family: georgia">
2682        * Set the excluded proposition to proposition <i>A</i>;
2683        * </font><BR>
2684        * myReferee.excludedProposition=A;<BR>
2685        * <font color="#0000FF" style="font-family: georgia">
2686        * Do the fusion of <i>aFuser1</i> and <i>aFuser2</i> into <i>aFuser</i>;
2687        * </font><BR>
2688        * aFuser.fuse(aFuser1,aFuser2,myReferee);
2689        * </code>
2690        *
2691        */
2692       public void Create_RefereeFunction() {
2693
```

```
2694          class myRefereeFunction<Prop extends ComplementedLattice<Prop> >
2695                        extends RefereeFunctionDempster<Prop> {
2696              public Prop excludedProposition = null;
2697
2698              @Override
2699              public ArrayList<Assignment<Prop>> refereeFunction(
2700                                      ArrayList<Assignment<Prop>> assignIn,
2701                                      ArrayList<minAssignment<Prop>>  bbaIn) {
2702                  if(excludedProposition==null) return super.refereeFunction(↩
                        assignIn, bbaIn);
2703                  ArrayList<Assignment<Prop>> output=super.refereeFunction(assignIn,↩
                         bbaIn);
2704                  if(excludedProposition.contains(output.get(0).attribute))
2705                      output.get(0).attribute.zero();
2706                  else output.get(0).attribute.and(output.get(0).attribute,
2707                                      excludedProposition.clone().cocomplement());
2708                  return output;
2709              }
2710
2711          }
2712
2713          finalPowerset A=new finalPowerset();
2714          A.size(3); A.atomic(0);
2715          finalPowerset B=A.instanceNsize(); B.atomic(1);
2716          finalPowerset C=A.instanceNsize(); C.atomic(2);
2717          //
2718          finalPowerset AUB=A.instanceNsize(); AUB.or(A,B);
2719          finalPowerset BUC=A.instanceNsize(); BUC.or(B,C);
2720          finalPowerset CUA=A.instanceNsize(); CUA.or(C,A);
2721          //
2722          finalPowerset zero=A.instanceNsize(); zero.zero();
2723          finalPowerset one=A.instanceNsize(); one.one();
2724          //
2725          //
2726          myRefereeFunction<finalPowerset> myReferee =
2727                              new myRefereeFunction<finalPowerset>();
2728          myReferee.excludedProposition=A;
2729          //
2730          //
2731          finalRefereeFuserRTS_Powerset aFuser1 = new finalRefereeFuserRTS_Powerset↩
                ();
2732          aFuser1.add(A,0.09);
2733          aFuser1.add(B,0.2);
2734          aFuser1.add(C,0.02);
2735          aFuser1.add(AUB,0.05);
2736          aFuser1.add(BUC,0.03);
2737          aFuser1.add(CUA,0.1);
2738          aFuser1.add(A,0.11);
2739          aFuser1.add(one,0.4);
2740          //
2741          finalRefereeFuserRTS_Powerset aFuser2 = aFuser1.instance();
2742          aFuser2.add(A,0.1);
2743          aFuser2.add(B,0.1);
2744          aFuser2.add(C,0.2);
2745          aFuser2.add(AUB,0.2);
2746          aFuser2.add(BUC,0.1);
2747          aFuser2.add(CUA,0.1);
2748          aFuser2.add(one,0.2);
2749          //
2750          finalRefereeFuserRTS_Powerset aFuser = aFuser1.instance();
2751  //
2752
2753          int printMode=1;
2754
2755          System.out.println(
2756                  "////////////////////////////\n" +
2757                  "// Relaxed Method\n" +
2758                  "////////////////////\n");
2759
2760          System.out.println("aFuser1");
2761              System.out.println(aFuser1.state(printMode));
2762              System.out.println();
2763          System.out.println("aFuser2");
2764              System.out.println(aFuser2.state(printMode));
```

```
2765                System.out.println();
2766            aFuser.fuse(aFuser1,aFuser2,myReferee);
2767            System.out.println("aFuser - myReferee");
2768                System.out.println("Conflict Z = "+aFuser.conflict()+" %");
2769                System.out.println(aFuser.state(printMode));
2770                System.out.println();
2771
2772        }
2773
2774
2775    }
```

Source Code: RefereeToolbox_Version.java

```
1  /*
2   *   RefereeToolbox_Version.java : part of package RefereeToolbox; Version.
3   ******************************************************************************
4   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
5   *
6   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
7   *
8   *   This file is part of RefereeToolbox.
9   *
10  *   RefereeToolbox is free software: you can redistribute it and/or modify
11  *   it under the terms of the GNU General Public License as published by
12  *   the Free Software Foundation, either version 3 of the License, or
13  *   (at your option) any later version.
14  *
15  *   RefereeToolbox is distributed in the hope that it will be useful,
16  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
17  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
18  *   GNU General Public License for more details.
19  *
20  *   You should have received a copy of the GNU General Public License
21  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
22  ******************************************************************************
23  */
24
25 package RefereeToolbox;
26
27 /**
28  * Class providing information about the version of package RefereeToolbox.
29  *
30  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
31  *
32  *
33  *
34  * <BR><BR><table border='1' cellPadding='4'><tr><td>
35  * <font color="#008000" style="font-family: georgia">
36  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
37  * <BR>
38  *   RefereeToolbox is free software: you can redistribute it and/or modify
39  *   it under the terms of the GNU General Public License as published by
40  *   the Free Software Foundation, either version 3 of the License, or
41  *   (at your option) any later version.<BR>
42  * <BR>
43  *   RefereeToolbox is distributed in the hope that it will be useful,
44  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
45  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
46  *   GNU General Public License for more details.<BR>
47  * <BR>
48  *   You should have received a copy of the GNU General Public License
49  *   along with RefereeToolbox. If not, see
50  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
51  * </font></table><BR>
52  *
53  */
54 public class RefereeToolbox_Version {
55
56 /**
57  * Return the current version name.
58  *
59  */
60     public String version() {
61         return "ZERO.0.1";
62     }
63
64 /**
65  * Return the current version code.
66  *
67  */
68     public int version_code() {
69         return 1;
70     }
71
```

```
72 }
```

Source Code: RelaxedBBA.java

```
1  /*
2   *   RelaxedBBA.java : part of package RefereeToolbox; Interface for Basic
3   *           Belief Assignment with relaxation.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Methods for relaxing Basic Belief Assignment. The relaxtion is based on a
30  * simple logarithmic-time summarization method.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public interface RelaxedBBA<Prop extends Lattice<Prop>, B extends RelaxedBBA<Prop,↩
        B> >
57                                         extends BasicBeliefAssignment<Prop,B> {
58
59 /**
60  * Relax <i>this</i> so that it contains no more than <i>maxMem</i> assignments.
61  *
62  */
63     boolean relax(int maxMem);
64
65 /**
66  * Relax <i>this</i> so that it contains no more than {@link RelaxedBBA#maxSize(↩
        int) }
67  * assignments.
68  *
69  */
```

```
70      boolean relax();
71
72  /**
73   * Set or return the default maximum number of assignments stored in <i>this</i>,
74   * depending on parameter <i>newSize</i>.
75   * If <i>newSize>0</i>, then set the default maximum number of assignments to <i>↩
         newSize</i>.
76   * In any case, return the actual value of the default maximum number.
77   * <BR><BR>
78   * N.B. It is not possible to exceed this number of assignments, except for ↩
         temporary
79   * computed assignments.
80   *
81   */
82      int maxSize(int newSize);
83  }
```

Source Code: RelaxedTreeSetBBA.java

```
1  /*
2   *   RelaxedTreeSetBBA.java : part of package RefereeToolbox; Implementation of
3   *          relaxed Basic Belief Assignment.
4   *****************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  *****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Relaxed extension of {@link TreeSetBBA }.
30  *
31  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
32  *
33  *
34  *
35  * <BR><BR><table border='1' cellPadding='4'><tr><td>
36  * <font color="#008000" style="font-family: georgia">
37  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
38  * <BR>
39  *   RefereeToolbox is free software: you can redistribute it and/or modify
40  *   it under the terms of the GNU General Public License as published by
41  *   the Free Software Foundation, either version 3 of the License, or
42  *   (at your option) any later version.<BR>
43  * <BR>
44  *   RefereeToolbox is distributed in the hope that it will be useful,
45  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
46  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
47  *   GNU General Public License for more details.<BR>
48  * <BR>
49  *   You should have received a copy of the GNU General Public License
50  *   along with RefereeToolbox. If not, see
51  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
52  * </font></table><BR>
53  *
54  */
55 public class RelaxedTreeSetBBA< Prop extends Lattice<Prop>,
56                                         B extends RelaxedTreeSetBBA<Prop,B> >
57                                            extends TreeSetBBA<Prop,B>
58                                            implements RelaxedBBA<Prop,B> {
59
60     @Override
61     protected boolean relaxed() { return true; }
62
63
64     ////////////////////////////////////////////////:
65     // public part
66     ///////////////
67
68 /**
69  * Relax <i>this</i> so that it contains no more than <i>maxMem</i> assignments.
70  *
71  */
```

```
72      @Override
73      public boolean relax(int maxMem) { return super.relax(maxMem); }
74
75  /**
76   * Relax <i>this</i> so that it contains no more than {@link RelaxedBBA#maxSize(↩
        int) }
77   * assignments.
78   *
79   */
80      @Override
81      public boolean relax() { return super.relax(); }
82
83  /**
84   * Set or return the default maximum number of assignments stored in <i>this</i>,
85   * depending on parameter <i>newSize</i>.
86   * If <i>newSize>0</i>, then set the default maximum number of assignments to <i>↩
        newSize</i>.
87   * In any case, return the actual value of the default maximum number.
88   * <BR><BR>
89   * N.B. It is not possible to exceed this number of assignments, except for ↩
        temporary
90   * computed assignments.
91   *
92   */
93      @Override
94      public int maxSize(int newSize) { return super.maxSize(newSize); }
95  }
```

Source Code: RFDempster_Powerset.java

```
1  /*
2   *   RFDempster_Powerset.java : part of package RefereeToolbox; Implementation
3   *         of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A non generic instance of {@link RefereeFunctionDempster } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RFDempster_Powerset extends RefereeFunctionDempster<finalPowerset> {
57
58 }
```

Source Code: RFDisjunctive_Powerset.java

```
1  /*
2   *   RFDisjunctive_Powerset.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A non generic instance of {@link RefereeFunctionDisjunctive } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RFDisjunctive_Powerset extends RefereeFunctionDisjunctive<↩
        finalPowerset> {
57
58 }
```

Source Code: RFDuboisPrade_Powerset.java

```
1  /*
2   *   RFDuboisPrade_Powerset.java : part of package RefereeToolbox;
3   *          Implementation of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A non generic instance of {@link RefereeFunctionDuboisPrade } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RFDuboisPrade_Powerset extends RefereeFunctionDuboisPrade<↩
        finalPowerset> {
57
58 }
```

Source Code: RFPCR6_Powerset.java

```
1  /*
2   *   RFPCR6_Powerset.java : part of package RefereeToolbox; Implementation of
3   *         referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A non generic instance of {@link RefereeFunctionPCR6 } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  *
32  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RFPCR6_Powerset extends RefereeFunctionPCR6<finalPowerset> {
57
58 }
```

Source Code: RFPCRSharp_Powerset.java

```
1  /*
2   *   RFPCRSharp_Powerset.java : part of package RefereeToolbox; Implementation
3   *         of referee function.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * A non generic instance of {@link RefereeFunctionPCRSharp } specialized for
30  * Lattice structures typed {@link finalPowerset }.
31  *
32  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
33  *
34  *
35  *
36  * <BR><BR><table border='1' cellPadding='4'><tr><td>
37  * <font color="#008000" style="font-family: georgia">
38  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
39  * <BR>
40  *   RefereeToolbox is free software: you can redistribute it and/or modify
41  *   it under the terms of the GNU General Public License as published by
42  *   the Free Software Foundation, either version 3 of the License, or
43  *   (at your option) any later version.<BR>
44  * <BR>
45  *   RefereeToolbox is distributed in the hope that it will be useful,
46  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
47  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
48  *   GNU General Public License for more details.<BR>
49  * <BR>
50  *   You should have received a copy of the GNU General Public License
51  *   along with RefereeToolbox. If not, see
52  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
53  * </font></table><BR>
54  *
55  */
56 public class RFPCRSharp_Powerset extends RefereeFunctionPCRSharp<finalPowerset> {
57
58 }
```

Source Code: SampledBBA.java

```
1  /*
2   *   SampledBBA.java : part of package RefereeToolbox; Interface for sampled
3   *          Basic Belief Assignment.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Methods for sampling Basic Belief Assignments and learning Basic Belief ↩
        Assignments
32  * from a samples vector.
33  *
34  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
35  *
36  *
37  *
38  * <BR><BR><table border='1' cellPadding='4'><tr><td>
39  * <font color="#008000" style="font-family: georgia">
40  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
41  * <BR>
42  *   RefereeToolbox is free software: you can redistribute it and/or modify
43  *   it under the terms of the GNU General Public License as published by
44  *   the Free Software Foundation, either version 3 of the License, or
45  *   (at your option) any later version.<BR>
46  * <BR>
47  *   RefereeToolbox is distributed in the hope that it will be useful,
48  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
49  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
50  *   GNU General Public License for more details.<BR>
51  * <BR>
52  *   You should have received a copy of the GNU General Public License
53  *   along with RefereeToolbox. If not, see
54  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
55  * </font></table><BR>
56  *
57  */
58 public interface SampledBBA<Prop extends Lattice<Prop>, sB extends SampledBBA<Prop↩
        ,sB> >
59                                          extends BasicBeliefAssignment<Prop,sB> {
60
61 /**
62  * Learn a new basic belief assignment for <i>this</i> from a given array of ↩
        weighted
63  * proposition particles. The weigthed particle are provided as an array of ↩
        assignment
64  * ArrayLattice<{@link Assignment }>.
65  * A mesure of conflict is returned, as a percentage of the conflicting samples.
66  *
67  */
```

```
68      double learnFrom(ArrayList<Assignment<Prop>> weightedSamples);
69
70  /**
71   * Make a proposition sample according to the basic belief assignement stored
72   * within <i>this</i>.
73   * The produced sample is weighted (at this time, the weight is 1), and, for this
74   * reason, is produced as an assignment {@link Assignment } with
75   * value 1.
76   *
77   */
78      Assignment<Prop> makeSample(); // weight may be null
79                  // in such a case, uniform weight is considered
80
81   /**
82   * Initialize the sampled mixer. Are needed as entry the basic belief assignments
83   * to be mixed, provided within array <i>bbaIn</i>, and the weights stored within
84   * <i>weights</i>. It is not necessary that the weights sum to 1, but the weights
85   * cannot be negative.
86   *
87   */
88      boolean setMixer(double[] weights, ArrayList<sB> bbaIn);
89
90  /**
91   * Make a <i>random</i> choice among the basic belief assignments in proportion
92   * to their respective weights.
93   * Both the weights and basic belief assignments have been first provided to ↩
       method
94   * {@link SampledBBARefereeFuser#setMixer }. It is necessary to run the method
95   * {@link SampledBBARefereeFuser#setMixer } before the first run of
96   * {@link SampledBBARefereeFuser#makeMixedChoice() }.
97   * The produced sample is weighted (at this time, the weight is 1), and, for this
98   * reason, is produced as an assignment {@link Assignment } with
99   * value 1.
100  *
101  */
102     sB makeMixedChoice();
103
104 }
```

Source Code: SampledBBARefereeFuser.java

```
1  /*
2   *   SampledBBARefereeFuser.java : part of package RefereeToolbox; Interface for
3   *            sampled fuser based on referee function.
4   ************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Methods for fusing Basic Belief Assignments by the means of referee functions
32  * and on the basis of a particle approximation. This interface does not concern
33  * direct rule implementations, {@link BBAFuser}, or exact referee-based fusion
34  * {@link BBARefereeFuser}.
35  *
36  * @see          BBAFuser
37  * @see          BBARefereeFuser
38  * @author       <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
      Dambreville</A>
39  *
40  *
41  *
42  * <BR><BR><table border='1' cellPadding='4'><tr><td>
43  * <font color="#008000" style="font-family: georgia">
44  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
45  * <BR>
46  *   RefereeToolbox is free software: you can redistribute it and/or modify
47  *   it under the terms of the GNU General Public License as published by
48  *   the Free Software Foundation, either version 3 of the License, or
49  *   (at your option) any later version.<BR>
50  * <BR>
51  *   RefereeToolbox is distributed in the hope that it will be useful,
52  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
53  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
54  *   GNU General Public License for more details.<BR>
55  * <BR>
56  *   You should have received a copy of the GNU General Public License
57  *   along with RefereeToolbox. If not, see
58  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
59  * </font></table><BR>
60  *
61  */
62 public interface SampledBBARefereeFuser<Prop extends Lattice<Prop>,
63                         sB extends SampledBBARefereeFuser<Prop,sB> >
64                     extends SampledBBA<Prop,sB> {
65
66 /**
67  * Initialize the sampled fuser. Are needed as entry the basic belief assignments
68  * to be fused, provided within array <i>bbaIn</i>, and a referee function
69  * <i>theRefereeFunction</i>.
70  *
71  */
```

```
72      boolean setFuser(ArrayList<sB> bbaIn, RefereeFunctionDefault<Prop> ↩
            theRefereeFunction);
73
74  /**
75   * Initialize the sampled fuser. Are needed as entry the basic belief assignments
76   * to be fused, provided as <i>left</i> and <i>right</i>, and a referee function
77   * <i>theRefereeFunction</i>.
78   *
79   */
80      boolean setFuser(sB left, sB right, RefereeFunctionDefault<Prop> ↩
            theRefereeFunction);
81
82  /**
83   * Make a proposition sample according to the fused basic belief assignement ↩
        implied
84   * by the basic belief assignements and referee function provided to method
85   * {@link SampledBBARefereeFuser#setFuser(java.util.ArrayList,
86   * RefereeToolbox.RefereeFunctionDefault) }
87   * or {@link SampledBBARefereeFuser#setFuser(RefereeToolbox.SampledBBARefereeFuser↩
        ,
88   * RefereeToolbox.SampledBBARefereeFuser, RefereeToolbox.RefereeFunctionDefault) ↩
        }.
89   * It is necessary to run these methods
90   * before the first run of
91   * {@link SampledBBARefereeFuser#makeFusedSample() }.
92   * The produced sample is weighted (at this time, the weight is 1), and, for this
93   * reason, is produced as an assignment {@link Assignment } with
94   * value 1.
95   *
96   */
97      Assignment<Prop> makeFusedSample();
98
99  }
```

Source Code: Superpowerset.java

```
1  /*
2   *   Superpowerset.java : part of package RefereeToolbox; Implementation of
3   *          superpowerset.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 /**
29  * Implementation of a Superpowerset structure by means of an array of {@code long}.
30  * Structural and logical  methods are inherited from {@link Freeboolean }.
31  * However, Superpowersets come with the constraint that the union of all atomic
32  * propositions is <i>one</i> or equivalently, the intersection of all negated atomic
33  * propositions is zero:
34  * <BR><BR>
35  * <i> <b>OR</b><sub>0 =< i < sizeFrame</sub> atomic(i) = one</i>
36  * <BR>
37  * <i> <b>AND</b><sub>0 =< i < sizeFrame</sub> atomic(i) = zero</i>
38  * <BR><BR>
39  * Then, <i>zero</i> is still defined by zeroing all bits; but <i>one</i> is defined
40  * by setting to 1 all active bits except the first one.
41  * The complement/cocomplement operators are inherited from the operators {@code ~}
42  *  working on {@code long}; exceeding bits are masked by a AND with <i>one</i>.
43  *
44  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric Dambreville</A>
45  *
46  *
47  *
48  * <BR><BR><table border='1' cellPadding='4'><tr><td>
49  * <font color="#008000" style="font-family: georgia">
50  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
51  * <BR>
52  *   RefereeToolbox is free software: you can redistribute it and/or modify
53  *   it under the terms of the GNU General Public License as published by
54  *   the Free Software Foundation, either version 3 of the License, or
55  *   (at your option) any later version.<BR>
56  * <BR>
57  *   RefereeToolbox is distributed in the hope that it will be useful,
58  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
59  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
60  *   GNU General Public License for more details.<BR>
61  * <BR>
62  *   You should have received a copy of the GNU General Public License
63  *   along with RefereeToolbox. If not, see
64  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
65  * </font></table><BR>
66  *
67  */
```

```
68  public class Superpowerset<L extends Superpowerset<L> > extends Freeboolean<L> {
69
70
71      ////////////////////////////////////////////////:
72      // public part
73      ///////////////
74
75  /**
76   * Set <i>this</i> to one, the neutral element for
77   * {@link Superpowerset#and(java.lang.Object, java.lang.Object) }.
78   * For the class {@link Superpowerset }, the first bit of <i>one</i> is set to 0.
79   *
80   */
81      @Override
82      public L one() {
83          super.one();
84          _memory[0]&=0xFFFFFFFFFFFFFFFEL; // remove the first bit
85          return (L) this;
86      }
87
88  /**
89   * Compute the <b>complement</b> of <i>aProposition</i> and store the result ↩
       within
90   * <i>this</i>.
91   * For the class {@link Superpowerset }, the first bit of the result is set to 0.
92   * <BR><BR>
93   * <b>Documentation inherited from {@link Freeboolean }:</b><BR>
94   * {@inheritDoc}
95   *
96   */
97      @Override
98      public L complement(L aProposition) {
99          super.complement(aProposition);
100         _memory[0]&=0xFFFFFFFFFFFFFFFEL; // remove the first bit
101         return (L) this;
102     }
103
104 /**
105  * Compute the <b>complement</b> of <i>this</i> and store the result within
106  * <i>this</i>.
107  * For the class {@link Superpowerset }, the first bit of the result is set to 0.
108  * <BR><BR>
109  * <b>Documentation inherited from {@link Freeboolean }:</b><BR>
110  * {@inheritDoc}
111  *
112  */
113     @Override
114     public L complement() {
115         super.complement();
116         _memory[0]&=0xFFFFFFFFFFFFFFFEL; // remove the first bit
117         return (L) this;
118     }
119
120 }
```

Source Code: TreeSetBBA.java

```
1  /*
2   *   TreeSetBBA.java : part of package RefereeToolbox; Implementation of Basic
3   *          Belief Assigment.
4   ******************************************************************************
5   *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *   Author:    FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *   This file is part of RefereeToolbox.
10  *
11  *   RefereeToolbox is free software: you can redistribute it and/or modify
12  *   it under the terms of the GNU General Public License as published by
13  *   the Free Software Foundation, either version 3 of the License, or
14  *   (at your option) any later version.
15  *
16  *   RefereeToolbox is distributed in the hope that it will be useful,
17  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *   GNU General Public License for more details.
20  *
21  *   You should have received a copy of the GNU General Public License
22  *   along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ******************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * Implement the Basic Belief Assigment by means of a double {@link TreeSet }.
32  * The TreeSet is applied to storing the list of assignments, typed {@link ↩
        Assignment },
33  * which define the Basic Belief Assigment. A double ordering of the assignment is
34  * implemented: an increasing total order over the attributes of the assignments,
35  * and a decreasing (total) ordering over the values of the assignments.
36  * The attribute-related ordering is instrumental for logarithmic manipulation of
37  * the assignments on the basis of an addressing by the attribute.
38  * The value-related ordering is used by subclasses of {@link TreeSetBBA} for
39  * relaxing the Basic Belief Assignment: by means of an interative summarization,
40  * each step being logarithmic time, the size of the assignments list is ↩
        maintained
41  * under a maximal bound.
42  * The relaxation is not available from this class.
43  * <BR>
44  * These logarithmic operations are made possible by the property of the class
45  * {@link TreeSet }.
46  *
47  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
48  *
49  *
50  *
51  * <BR><BR><table border='1' cellPadding='4'><tr><td>
52  * <font color="#008000" style="font-family: georgia">
53  *   Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
54  * <BR>
55  *   RefereeToolbox is free software: you can redistribute it and/or modify
56  *   it under the terms of the GNU General Public License as published by
57  *   the Free Software Foundation, either version 3 of the License, or
58  *   (at your option) any later version.<BR>
59  * <BR>
60  *   RefereeToolbox is distributed in the hope that it will be useful,
61  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
62  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
63  *   GNU General Public License for more details.<BR>
64  * <BR>
65  *   You should have received a copy of the GNU General Public License
66  *   along with RefereeToolbox. If not, see
67  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
68  * </font></table><BR>
69  *
```

```
70  */
71 public class TreeSetBBA<Prop extends Lattice<Prop>,
72                                             B extends TreeSetBBA<Prop,B> >
73                                              extends BBACommon<Prop,B> {
74
75     protected int theMaxSize = 20;
76
77     protected boolean relaxed() { return false; }
78
79     protected final Comparator< Assignment<Prop> > comparatorA =
80             new Comparator< Assignment<Prop> > () {
81         //
82         public int compare(Assignment<Prop> left, Assignment<Prop> right) {
83
84             return left.attribute.compareTo(right.attribute);
85         }
86     };
87
88
89     protected final Comparator< Assignment<Prop> > comparatorVA =
90             new Comparator< Assignment<Prop> > () {
91
92         public int compare(Assignment<Prop> left, Assignment<Prop> right) {
93             // Comparison order is reverted
94             int answerV =  right.value.compareTo(left.value);
95             if(answerV!=0) return answerV;
96             return left.attribute.compareTo(right.attribute);
97         }
98     };
99
100
101    protected TreeSet<Assignment<Prop>> bbaTreeA = new TreeSet<Assignment<Prop>>(↩
        comparatorA);
102
103    protected TreeSet<Assignment<Prop>> bbaTreeVA = new TreeSet<Assignment<Prop>>(↩
        comparatorVA);
104
105
106    protected boolean _relaxStep() { // only used with the relaxed extensions of ↩
        the class
107        if(bbaTreeA.size()<2) {
108            System.err.println("Error :: "+this.getClass().getName() +
109                ".relaxStep() :: bbaTreeA.size()<2");
110            System.exit(0);
111        }
112        Assignment<Prop> lastPair=bbaTreeVA.pollLast();
113        Assignment<Prop> prevLastPair = bbaTreeVA.pollLast();
114        bbaTreeA.remove(lastPair);
115        bbaTreeA.remove(prevLastPair);
116        prevLastPair.value+=lastPair.value;
117        prevLastPair.attribute.or(prevLastPair.attribute,lastPair.attribute);
118        add(prevLastPair);
119        return true;
120    }
121
122
123    ////////////////////////////////////////////////:
124    // public part
125    ///////////////
126
127 /**
128  * Do an exact copy of input into <i>this</i>.
129  *
130  */
131    @Override
132    public B duplicate(B input) {
133        load(input.bbaTreeVA);
134        return (B) this;
135    }
136
137 /**
138  * Return a representation of the state of <i>this</i> printed as
139  * a String. Typically, this state is the list of all stored assignments (↩
        proposition
```

```
140  * and value). Depending on the parameter <i>choix</i> in entry, the ↩
         representation
141  * is modified as folllows:<br><br>
142  * <i>choix==0</i>  --  Nothing is printed.
143  * <br>
144  *<i>choix==1</i>  --  Assignments are printed in increasing order of their
145  * propositions,
146  * <br>
147  *<i>choix==2</i>  --  Assignments are printed in decreasing order of their
148  * value,
149  * <br>
150  *<i>choix==3</i>  --  Do both print in that order.
151  * <br><br>
152  * Assignments are printed according to the following format:
153  * <br>
154  * <code> attribute -> value </code>
155  * <br>
156  * The attribute is printed according to its own state method {@link Lattice#state↩
         () }.
157  *
158  */
159     @Override
160     public String state(int choix) {
161         String theState="";
162         if((choix&1)==1) {
163             theState="bbaTreeA:\n";
164             for (Iterator<Assignment<Prop>> it = bbaTreeA.iterator();
165                                                         it.hasNext(); ) {
166                 Assignment<Prop> currentPair=it.next();
167                 theState+=currentPair.attribute.state() + " -> " +
168                                                 currentPair.value + "\n";
169             }
170         }
171         if((choix&3)==3) theState+="\n";
172         if((choix&2)==2) {
173             theState+="bbaTreeVA:\n";
174             for (Iterator<Assignment<Prop>> it = bbaTreeVA.iterator();
175                                                         it.hasNext(); ) {
176                 Assignment<Prop> currentPair=it.next();
177                 theState+=currentPair.attribute.state() + " -> " +
178                                                 currentPair.value + "\n";
179             }
180         }
181         return theState;
182     }
183
184
185 /**
186  * Create an array of the assignments stored within the class. These assignments ↩
         not
187  * necessary sum to 1. Notice that the assignments should be positively valued
188  * (focal elements), although <i>this</i> is not a strict requirement.
189  *
190  */
191     @Override
192     public ArrayList<Assignment<Prop>> toArray() {
193         return new ArrayList<Assignment<Prop>> (bbaTreeVA);
194     }
195
196 /**
197  * Remove all assignments from <i>this</i>. The class instance is cleared.
198  *
199  */
200     @Override
201     public void clear() {
202         bbaTreeA.clear();
203         bbaTreeVA.clear();
204         update_notification();
205     }
206
207 /**
208  * Add an assignment characterized by a proposition <i>aProposition</i> and a ↩
         value
209  * <i>anAssignmentValue</i> to <i>this</i>.
```

```
210  * If an assignment already exists for <i>aProposition</i>, say with value <i>theOldValue</i>,
211  * then the new assignement of <i>aProposition</i> is stored with value
212  * <i>theOldValue+anAssignmentValue</i>.
213  *
214  */
215     @Override
216     public boolean add(Prop aProposition, double anAssignmentValue) {
217         if(anAssignmentValue<=0) return false;
218         Assignment<Prop> anAssignment=new Assignment<Prop>();
219         anAssignment.attribute=aProposition.clone(); // avoid border effect with entry
220         anAssignment.value=anAssignmentValue;
221         Assignment<Prop>  foundPair=findProposition(anAssignment);
222         if(foundPair!=null) {
223                 anAssignment.value+= foundPair.value;
224                 bbaTreeA.remove(foundPair);
225                 bbaTreeVA.remove(foundPair);
226         }
227         bbaTreeA.add(anAssignment);
228         bbaTreeVA.add(anAssignment);
229         update_notification();
230         return true;
231     }
232 
233 /**
234  * Search if there is an assignment stored within the class, and which attribute
235  * is equal to the attribute of <i>anAssignment</i>.
236  * Return the found assignment, if there is one. Othewise, return <i>null</i>.
237  *
238  */
239     @Override
240     public Assignment<Prop> findProposition(Assignment<Prop> anAssignment) {
241         Assignment<Prop>  foundPair=bbaTreeA.ceiling(anAssignment);
242         if(foundPair!=null) {
243             if(foundPair.attribute.compareTo(anAssignment.attribute)==0) {
244                 return foundPair;
245             }
246         }
247         return null;
248     }
249 
250 /**
251  * Get the belief of <i>aProposition</i>. Although not required, it is assumed that
252  * the assignments sum to 1.
253  *
254  */
255     @Override
256     public double Bel(Prop aProposition) {
257         double belief=0.;
258         for (Iterator<Assignment<Prop>> it = bbaTreeA.iterator();
259                                             it.hasNext(); ) {
260             Assignment<Prop> currentPair=it.next();
261             if(aProposition.contains(currentPair.attribute))
262                                             belief+=currentPair.value;
263         }
264         return belief;
265     }
266 
267 /**
268  * Get the plausibility of <i>aProposition</i>. Although not required, it is assumed that
269  * the assignments sum to 1.
270  *
271  */
272     @Override
273     public double Pl(Prop aProposition) {
274         double belief=0.;
275         for (Iterator<Assignment<Prop>> it = bbaTreeA.iterator();
276                                             it.hasNext(); ) {
277             Assignment<Prop> currentPair=it.next();
278             if(aProposition.intersects(currentPair.attribute))
279                                             belief+=currentPair.value;
```

```
280         }
281         return belief;
282     }
283 
284 /**
285  * Remove the assignment related to proposition <i>aProposition</i>, if there is
286  * such assignment stored within <i>this</i>.
287  * Return the value assigned to the proposition.
288  *
289  */
290     @Override
291     public double remove(Prop aProposition) {
292         Assignment<Prop>  foundPair=findProposition(aProposition);
293         if(foundPair!=null) {
294                 double theValue=foundPair.value;
295                 bbaTreeA.remove(foundPair);
296                 bbaTreeVA.remove(foundPair);
297                 update_notification();
298                 return theValue;
299             }
300         return 0.;
301     }
302 
303 /**
304  * Make a mix of the basic belief assignments stored within the entry, <i>bbaIn</i>
305  * according to their respective weight <i>weight</i>. This mix is stored into <i>this</i>.
306  *
307  */
308     @Override
309     public B mix(ArrayList<B> bbaIn, double[] weight) {
310         int nbIn=bbaIn.size();
311 
312         if(nbIn<1) return null;
313         if(nbIn!=weight.length) return null;
314         double cumul=0;
315         double aValue;
316         int i;
317         Iterator j;
318         Assignment<Prop> refOut;
319         for(i=0;i<nbIn;i++) {
320             aValue=weight[i];
321             if(aValue<0) return null;
322             cumul+=aValue;
323         }
324         if(cumul==0) return null;
325         clear();
326         B bbaResult=instance();
327         for(i=0;i<nbIn;i++) {
328             aValue=weight[i]/cumul;
329             for(j=(bbaIn.get(i)).bbaTreeVA.iterator(); j.hasNext();) {
330                 refOut=(Assignment<Prop>) j.next();
331                 bbaResult.add(refOut.attribute, aValue*refOut.value);
332             }
333         }
334         duplicate(bbaResult);
335         update_notification();
336         return (B) this;
337     }
338 
339     @Override
340     public void update_notification() { // called each time a change is done
341         if(relaxed()) relax();
342         else super.update_notification();
343     }
344 
345 
346     ////////////////////////////////////////////////:
347     // reserved part
348     ///////////////
349 
350 /**
351  * This method is not accessible from class {@link TreeSetBBA } and generates an
```

```
     error
352  * report and a code exit.
353  *
354  */
355     public boolean relax(int maxMem) {
356         if(!relaxed()) {
357             System.err.println("Class" + this.getClass().getName()
358                                        + " does not implement relaxation");
359             System.exit(0);
360         }
361         if(maxMem<1) return false;
362         boolean test=true;
363         while(bbaTreeA.size()>maxMem) {
364             test=test&&_relaxStep();
365         }
366         super.update_notification();
367         return test;
368     }
369 
370 /**
371  * This method is not accessible from class {@link TreeSetBBA } and generates an ↩
     error
372  * report and a code exit.
373  *
374  */
375     public boolean relax(){
376         if(!relaxed()) {
377             System.err.println("Class" + this.getClass().getName()
378                                        + " does not implement relaxation");
379             System.exit(0);
380         }
381         return relax(maxSize(-1));
382     }
383 
384 /**
385  * This method is not accessible from class {@link TreeSetBBA } and generates an ↩
     error
386  * report and a code exit.
387  *
388  */
389     public int maxSize(int newSize) {
390         if(!relaxed()) {
391             System.err.println("Class" + this.getClass().getName()
392                                        + " does not implement relaxation");
393             System.exit(0);
394         }
395         if(newSize>0) {
396             theMaxSize=newSize;
397             relax();
398         }
399         return theMaxSize;
400     }
401 
402 }
```

Source Code: TreeSetRefereeFuser.java

```
1  /*
2   *    TreeSetRefereeFuser.java : part of package RefereeToolbox; Implementation
3   *          of fuser based on referee function.
4   ****************************************************************************
5   *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville
6   *
7   *    Author:     FrÃ©dÃ©ric Dambreville <http://email.fredericdambreville.com>
8   *
9   *    This file is part of RefereeToolbox.
10  *
11  *    RefereeToolbox is free software: you can redistribute it and/or modify
12  *    it under the terms of the GNU General Public License as published by
13  *    the Free Software Foundation, either version 3 of the License, or
14  *    (at your option) any later version.
15  *
16  *    RefereeToolbox is distributed in the hope that it will be useful,
17  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
18  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
19  *    GNU General Public License for more details.
20  *
21  *    You should have received a copy of the GNU General Public License
22  *    along with RefereeToolbox. If not, see <http://www.gnu.org/licenses/>.
23  ****************************************************************************
24  */
25
26 package RefereeToolbox;
27
28 import java.util.*;
29
30 /**
31  * This class implements a generic combination of Basic Belief Assignments by the
32  * means of referee functions and on the basis of an exact computation. This
33  * class does not handle direct rule implementations, {@link BBAFuser}, or
34  * implementations based on referee sampling, {@link SampledBBARefereeFuser}.
35  *
36  * @author      <A href="http://email.fredericdambreville.com">FrÃ©dÃ©ric ↩
        Dambreville</A>
37  *
38  *
39  *
40  * <BR><BR><table border='1' cellPadding='4'><tr><td>
41  * <font color="#008000" style="font-family: georgia">
42  *    Copyright (c) 2010 FrÃ©dÃ©ric Dambreville<BR>
43  * <BR>
44  *    RefereeToolbox is free software: you can redistribute it and/or modify
45  *    it under the terms of the GNU General Public License as published by
46  *    the Free Software Foundation, either version 3 of the License, or
47  *    (at your option) any later version.<BR>
48  * <BR>
49  *    RefereeToolbox is distributed in the hope that it will be useful,
50  *    but WITHOUT ANY WARRANTY; without even the implied warranty of
51  *    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
52  *    GNU General Public License for more details.<BR>
53  * <BR>
54  *    You should have received a copy of the GNU General Public License
55  *    along with RefereeToolbox. If not, see
56  * <a href='http://www.gnu.org/licenses/'>http://www.gnu.org/licenses/</a>.
57  * </font></table><BR>
58  *
59  */
60
61 public class TreeSetRefereeFuser<Prop extends Lattice<Prop>,
62             B extends TreeSetRefereeFuser<Prop,B>  >
63         extends TreeSetBBA<Prop,B>
64         implements BBARefereeFuser<Prop, B> {
65
66     protected double theConflict=-1.;
67     protected Prop theZero=null;
68     protected ArrayList<minAssignment<Prop>> BBASources= new ArrayList<↩
          minAssignment<Prop>>();
69
70
```

```
71      ////////////////////////////////////////////////:
72      // public part
73      ///////////////
74
75      public double conflict() {
76          return theConflict;
77      }
78
79      public B fuse(B left, B right, RefereeFunctionDefault<Prop> theRefereeFunction↩
            ) {
80          ArrayList<B> bbaTab= new ArrayList<B>();
81          bbaTab.add(left);
82          bbaTab.add(right);
83          return fuse(bbaTab, theRefereeFunction);
84      }
85
86
87      public B fuse(ArrayList<B> bbaIn, RefereeFunctionDefault<Prop> ↩
            theRefereeFunction) {
88          int nbIn;
89          int k,l;
90          double Norm;
91          nbIn=bbaIn.size();
92          if(nbIn<1) return null;
93          Iterator[] j=new Iterator[nbIn];
94          ArrayList<Assignment<Prop>> assignIn = new ArrayList<Assignment<Prop>>();
95          BBASources.clear();
96          for(k=0;k<nbIn;k++) {
97              assignIn.add(null);
98              BBASources.add(bbaIn.get(k));
99          } // Chercher mieux ?
100         //
101         double[] propEval= new double[1+nbIn];
102         propEval[nbIn]=1.;
103         ArrayList<Assignment<Prop>> refOut;
104         Assignment<Prop> aTmpAssign;
105         B bbaResult= instance();
106         bbaResult.clear();
107         for(k=0;k<nbIn;k++) j[k]=bbaIn.get(k).bbaTreeVA.iterator();
108         for(k=nbIn-1;k<nbIn;) {
109             if(j[k].hasNext()) {
110                 aTmpAssign=(Assignment<Prop>) j[k].next();
111                 assignIn.set(k, aTmpAssign);
112                 propEval[k]=propEval[k+1]*aTmpAssign.value;
113                 if(k==0) {
114                     refOut=theRefereeFunction.refereeFunction(assignIn,BBASources)↩
                            ;
115                     for(l=0;l<refOut.size();l++)
116                         bbaResult.add(refOut.get(l).attribute,
117                                       propEval[0]*refOut.get(l).value);
118                 } else {
119                     k--;
120                 }
121             } else {
122                 j[k]=bbaIn.get(k).bbaTreeVA.iterator();
123                 k++;
124             }
125         }
126         bbaResult.update_notification();
127         Norm=0.;
128         if(theZero==null)
129                 theZero=bbaResult.bbaTreeA.first().attribute.instanceNsize().↩
                        zero();
130         theConflict=bbaResult.remove(theZero);
131         for (Iterator<Assignment<Prop>> it = bbaResult.bbaTreeA.iterator();
132                                             it.hasNext(); ) {
133             Assignment<Prop> currentPair=it.next();
134             Norm+=currentPair.value;
135         }
136         clear();
137         for (Iterator<Assignment<Prop>> it = bbaResult.bbaTreeA.iterator();
138                                             it.hasNext(); ) {
139             Assignment<Prop> currentPair=it.next();
140             currentPair.value/=Norm;
```

```
141                 add(currentPair);
142             }
143             update_notification();
144 
145             return (B) this;
146         }
147 
148 }
```

4 License

RefereeToolbox is released under the GNU GENERAL PUBLIC LICENSE:

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <\protect\vrule width0pt\protect\href{http:/
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                              Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
```

changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the

public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions.

Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a

patent against the party.

  If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

  A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may

not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM

IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

```
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <\protect\vrule width0pt\protect\href{http://www.gnu.
```

```
Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License.  Of course, your program’s commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<\protect\vrule width0pt\protect\href{http://www.gnu.org/licenses/}{http://www.gnu.org/licenses
```

```
  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<\protect\vrule width0pt\protect\href{http://www.gnu.org/philosophy/why-not-lgpl.html}{http://w
```