(2)

An Evidential Probability Shell User's Manual for version 1.00 **Draft***

Bülent Murtezaoğlu
mucit@cs.rochester.edu
Henry E. Kyburg
kyburg@cs.rochester.edu

November 20, 1990

*Research underlying this work has been supported in part by the Signals Warfare Center of the U.S. Army.

REPORT DOCUMENTATION PAGE

Form Approved OPM No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE 1990	3. REPORT TYPE AND DATES COVERED Unknown

4. TITLE AND SUBTITLE

An Evidential Probability Shell User's Manual for version 1.00

5. FUNDING NUMBERS

DAAB10-86-C-0567

6. AUTHOR(S)

Henry E. Kyburg, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Rochester
Department of Philosophy
Rochester, NY 14627

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U.S. Army CECOM Signals Warfare Directorate
Vint Hill Farms Station
Warrenton, VA 22186-5100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

92-TRF-0001

1. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Statement A; Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

AEPS is a research tool designed for investigating various theoretical and practical issues concerning the statistical inference procedures developed by Henry Kyburg. Since research about both the theoretical and the practical aspects of the system is active and evolving, more powerful versions of this program will eventually become available. The motivation for implementing the procedure was to test hands on how various modifications would affect the outcomes of the inferences. We have in the course of writing and testing the code changed our minds about the general applicability of the XP style evidence combination method mainly because of the clear-cut counter examples we could come up with [30]. We have implemented several variants of the procedure for experimentation and to gain more insight into what might be both intuitively appealing and formally elegant (not to mention computationally efficient). Having all the variants of the methods built into an interactive shell that uses a simple syntax side by side comparison possible.

1. SUBJECT TERMS Artificial Intelligence, Data Fusion, Evidential Probability Philosophy	15. NUMBER OF PAGES 27
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Introduction

AEPS is a research tool designed for investigating various theoretical and practical issues concerning the statistical inference procedure developed by Henry Kyburg [2]. A subset of the procedure, limited to the homogeneous case, has been implemented in this version. The methods implemented include the core of the ones discussed in [5] [4] [3] as well as some new variants [1].

Since research about both the theoretical and the practical aspects of the system is active and evolving, more powerful versions of this program will eventually become available.

The motivation for implementing the procedure was to test hands on how various modifications would affect the outcomes of the inferences. We have in the course of writing and testing the code changed our minds about the general applicability of the XP style evidence combination method mainly because of the clear-cut counter examples we could come up with [3][1]. We have implemented several variants of the procedure for experimentation and to gain more insight into what might be both intuitively appealing and formally elegant (not to mention computationally efficient). Having all the variants of the methods built into an interactive shell that uses a simple syntax made side by side comparison possible.

We believe the advantage of having an easy to use shell is twofold. One is educational, it is much easier for students and researchers to understand how Kyburg's procedure works when they have a tool that they can play with. The second advantage is that it enables other researchers to investigate the utility of the system as part of a solution to bigger problems, such as expert systems, procedures for solving the frame problem, object recognition and others.

The shell is not meant to be used as an efficient special purpose inference engine; rather, it is meant to be used to gain insight into what procedure yields useful results and what should be implemented and embedded in other systems.

Overview

The subset of Kyburg's procedure that this shell implements can be summarized in two parts.

The Knowledge Base

Although Kyburg specifies a method for finding a non-trivial probability interval for any sentence as long as his knowledge base contains enough information to compute the interval for a logically equivalent sentence, the system of this paper completely by-passes this step by requiring the query to be in a convenient set membership form. Even with this simplification, the resulting complexity problems are not trivial.

The knowledge base (henceforth $\mathcal{K}$) contains:

- Set membership statements of the form

  "$x \in Y$" where x is an object and Y is a set.

- Subset relationship statements of the form

  "$Y \subset Z$"

  where Y and Z are sets.

- Statements concerning proportions of sets of the form

  "$\%(P, S) \in [p, q]$" where P and S are sets and p and q are some approximate representation of real numbers. These can be read as " the measure or the proportion of elements of set S that have the property P is in the interval $[p, q]$."

The sets in membership and subset statements and the set S in the % statement can be an intersection of sets. Neither set negations nor unions of sets are allowed.

Methods

Given a query "Prob(x,P)" (to be read "what is the "probability" of object[1] x having the property P?") the interval $[p, q]$ is returned if and only if there exist Y and such that

- "$x \in Y$" is entailed[2] by $\mathcal{K}$.
- "$\%(P, Y) \in [p, q]$" is entailed by $\mathcal{K}$.
- Y is *the right reference class* for x given $\mathcal{K}$.

The methods we will describe specify how the right reference class is to be selected and what exactly is entailed by the knowledge base.

First, a definition: an inference structure (henceforth IS) for a query "Prob(x,P)" is a quintuple

$$< x, P, S, p, q >$$

where

- S is a set.
- "$x \in S$" is entailed by $\mathcal{K}$.
- "$\%(P, S) \in [p, q]$" is entailed $\mathcal{K}$.

"$x \in S$" can be entailed by the knowledge base in one or a combination of the following ways:

- " $x \in S$ " $\in \mathcal{K}$.
- " $x \in S_1$ " $\in \mathcal{K} \wedge$" $S_n \subset S$ " $\in \mathcal{K} \wedge \cdots \wedge$ " $S_n \subset S$ " $\in \mathcal{K}$.

[1]x does not have to be a single object; it can be a set of objects or a set consisting of a single object. I will talk about objects throughout this paper.

[2]I use the term "entailed by $\mathcal{K}$" rather than the usual "in $\mathcal{K}$" because I use $\mathcal{K}$ to denote the set of pieces of knowledge we are explicitly given.

- If " $x \in S_1 \cap \ldots \cap S_n$ " is entailed by $\mathcal{K}$, then " $x \in S_1$ ", ... ," $x \in S_n$ " are also entailed by $\mathcal{K}$.

It should be clear from the definitions that while there can be many such IS for a given query and a knowledge base, their number is conveniently bounded by how much we know about the world (or equivalently how many "%" statements, subset statements and intersection statements are explicitly given).

Methods

Four different inference procedures have been implemented in this version of the program. The first three (methods 1 to 3) are variants of the procedure outlined in [6] [1], and one (method 4) is a new method [3] [1].

The XP Construction

The XP construction (due to R. Loui [5]) is a way of combining evidence. It enables the system to make useful inferences using every bit of relevant knowledge, but the results it yields are not always intuitive and it has inherent computational complexity problems. It is an addition to the entailment rules outlined above. Formally, we say:

Whenever $\mathcal{K}$ entails inference structures IS_1 and IS_2 of the form

$$IS_1 = < x, P, R_1, p_1, q_1 >$$

$$IS_2 = < x, P, R_2, p_2, q_2 >$$

it also entails

$$IS_3 = << x, x >, < P, P >, XP_P(R_1, R_2), g(p_1, p_2), g(q_1, q_2) >$$

where

$$g(x, y) = \frac{xy}{1 - x - y + 2xy}$$

It should be noted that there are $O(2^n)$ XP style inference structures for n relevant inference structures.

Conflicts and Dominance Rules

Only the definitions for conflicts and dominance rules between IS will be given in this document; the discussion and justification for the rules can be found elsewhere [2] [6] [1].

A conflict between two candidate inference structures is said to arise whenever their associated intervals do not nest within each other. To resolve conflicts we use two rules, one for the case where XP's are not involved and one for the case where they are [3].

[3]Actually the rule that is used for the XP's is a more powerful version of the one used for the non-XP case [1].

Dominance Between non-XP IS

The subset principle [2] states that when two inference structures $IS_1 = < x, P, R_1, p_1, q_1 >$ and $IS_2 = < x, P, R_2, p_2, q_2 >$ disagree, and if $\mathcal{K}$ entails additional knowledge stating $R_1 \subset R_2$, then we disregard IS_2. In this case IS_1 is said to dominate IS_2.

0.0.1 Dominance with XP's

When two XP style IS

$IS_a =<< x, \ldots, x >, P\times, \cdots, \times P, XP_P(R_{a_1}, \ldots, R_{a_n}), p_a, q_a >$

$IS_b =<< x, \ldots, x >, P\times, \cdots, \times P, XP_P(R_{b_1}, \ldots, R_{b_m}), p_b, q_b >$

conflict, and if $\mathcal{K}$ entails additional knowledge stating that for each R_{b_i} there is an R_{a_j} such that $R_{a_j} \subseteq R_{b_i}$, then IS_a dominates IS_b. It should be noted that the subset principle is the special case where the XP style set has just one constituent.

Selecting The Reference Class

Once the conflicts are settled, all that remains is a set of IS, no two of which disagree. The IS with the strongest interval from this set is selected as the IS for the reference class [1] [2]. Given that the initial set of candidate non-trivial inference structures is non-empty, and the XP construction is used to combine IS, such a reference class is guaranteed to exist [1] and can be found using an $O(n^2)$ algorithm [6] [1].

The methods we have implemented differ in the way they approach the issue of the entailment of the XP style inference structures. As can be seen from the example outputs given in this document, the XP construction does not always yield intuitive results. Some modifications to the entailment rules, some of which border on heuristics [1], have been tried in this implementation.

Method 1

This method is identical to the one described in [6] and [4]. Given a query, method 1 proceeds as follows:

1. All the relevant non-trivial IS [4], entailed by the knowledge base through subset chaining and expansion of intersections are found.
2. The XP style inference structures are constructed using the IS with no known subset relationships between them. [5]
3. The reference class is selected from the set of IS constructed in 1 and 2.

Steps 1 and 2 together take $O(2^n)$ space and time where n is the number of IS found in step 1. Step 3 takes $O(n^2)$ time, yielding an overall time complexity of $O(2^{2n})$ and space complexity of $O(2^n)$.

[4]With intervals narrower than [0, 1].

[5]Note that this cuts down the number of XP style inference structures somewhat.

Method 2

This is an attempt to cut down the number of XP style IS [6]. All conflicts that can be resolved by the subset principle are resolved and the dominated inference structures are deleted before the XP's are constructed. Since XP's are constructed using only the conflicting IS with no subset relationships between them, the computation terminates in a shorter time in most cases.

For this method and the following ones, the subset principle has to be slightly modified to make sure that only those IS which dominate over all the others they conflict with are saved [1]. The outline of method 2 is as follows:

1. All the relevant non-trivial IS entailed by the knowledge base through subset chaining and expansion of intersections are found.
2. Conflicts between the candidate reference classes found in 1 are settled in the following way:

   An inference structure is excluded from the set of candidates before the XP's are constructed, if

   - it conflicts with and is dominated by another or
   - all the candidates it conflicts with are excluded by the first rule.
3. All the XP's are constructed using the set of candidates obtained in 2.
4. The reference class is selected from the set of candidates constructed in 3.

While the worst case time complexity of this method is slightly worse than method 1 (due to the additional computation in step 2), it is much faster when subset relationships between candidates are known [7].

Method 3

This method is identical to method 2 except for an additional rule for exclusion:

An inference structure is excluded from the set of candidates before XP's are constructed if there is a stronger (narrower interval) one which agrees with it.

This is the fastest method using XP's. It should be noted that, by limiting the precision of the endpoints of intervals, the search time can be bounded independent of the size of the set of possible inference structures.

Method 4

This one is the fastest of the methods. The computation is identical to method 3, except that instead of constructing the XP's, it constructs the interval cover of the remaining candidates after some are excluded by the above rules.

[6] For a discussion of this method see [1].

[7] The time complexity decreases by a factor of 4 whenever a candidate IS is disregarded.

The Syntax

A fully parenthesized prefix language has been implemented. Since the standard Common LISP function *read* is used as the input routine, minimal exposure to LISP would be sufficient for feeling comfortable with the syntax.

The input to the system is always a list. Comments starting with a ";" are allowed and disregarded both in the interactive mode and when input files are used.

Startup and Exit

A session is started by loading the file *aeps.lisp* into the Common LISP interpreter. If for any reason the execution of the program is interrupted and the control returns to the lisp top level, invoking the function *start-shell* should start the shell from scratch [8].

Typing the command *(bye)* at the shell prompt causes the program to exit.

Shell Variables

Shell variables are used to hold the values for various parameters needed by the inference engine. The values of shell variables and switches are set using the *set* command.

EG: (set confidence .9)

The shell variables used in this version are:

- **Confidence** The confidence level used to convert the sampling data to confidence intervals. Its default value is 0.95. Whenever a new confidence level is set, all the intervals corresponding to sampling data are re-computed.
- **Method** This is the variable to set to change the method used to compute the reference class. Currently four values (1 to 4) are possible. A brief description of each method is given below. The default method is 4.
- **Precision** The number of decimal places taken as significant when testing for conflicts. It is set to 4 by default.
- **Trace** This flag is used to tell the program to print tracing information like lists of candidate reference classes and conflicts between them at various stages of computation. The permissible values are *on* and *off*. The default is *on*.
- **Time** When this flag is set, the program prints the timing information about the execution of queries. This information is acquired by a call to the Common LISP function *time*. The information printed may vary among Common LISP implementations. The default is *on*.

[8]It is possible to crash the input routine and get dropped into the lisp debugger by inputting commas etc. that *read* does not expect. Restarting the shell might be necessary in those cases. Anything else that invokes the debugger is probably an unknown bug.

- **xp-limit** Limits the number of XP style inference structures that the program constructs. This limit depends on your LISP implementation and your patience. For small data files, all the possible XP's will be constructed in a reasonable amount of time; for large files this limit may be reached before XP generation is completed. The default is 62350.

Data Input Commands

Three Kinds of data may be input:

1. Interval data of the form

   $$(\% \, (< property > < set >) \, (p \; q))$$

   meaning that the proportion of the number of elements of the $< set >$ with $< property >$ is known to be in the interval $[p, q]$.

   EG: (% (female human) (0.49 0.52))

   Internally, % statements are interpreted as valid under any confidence level, so it is a good idea to give the program sampling data rather than probability data whenever possible. This shortcoming may be corrected in future versions of the program.

2. Sampling data of the form

   $$(s\% \, (< property > < set >) \, (t \; s))$$

   tell the program that out of t trials on members of $< set >$ s members have been discovered to have the $< property >$. Sampling data given by s% statements are converted to interval data by approximating binomial confidence intervals at the current level of confidence.

   EG: (s% (male student) (4500 3000))

3. Subset relationship data of the form

   $$(subs \; < set > \; (< set_1 > < set_2 > \; \ldots \; < set_n >))$$

   meaning that $< set_1 >, < set_2 >, \ldots, < set_n >$ are supersets of $< set >$.

   EG: (subs cows (mammals animals))

   The identical syntax is used for set membership statements:

   $$(mem \; < object > \; (< set_1 > < set_2 > \ldots < set_n >))$$

   EG: (mem tweety (bird penguin))

Notation for Sets

Any legal lisp symbol or number, except the reserved symbols **I, XP, OR,** may be used to denote sets. Set negation and unions are not permitted; set intersections are permitted. An intersection of sets is denoted by the list

$$(\mathbf{I} < set_1 > < set_2 > \ldots < set_n >)$$

.

Queries

The syntax for queries is:

$$(prob(< property >< object >))$$

to be read: "what is the probability of the $< object >$ having the $< property >$?."

EG: (prob (flyer tweety))

Miscellaneous Commands

Clear: clears the knowledge base. All the sampling, interval, subset and membership data are erased. This command is intended to enable the user to experiment with different sets of evidence within the same session.

Reset: completely resets the shell. Does a *clear* and sets the shell variables to their default values.

Load: loads an input file. The syntax is :

(load "file-name")

If tracing is on, the input is echoed to the standard output as it is read from the file. The prompt changes to *loading* ==> to avoid confusion.

Nesting of load commands is not allowed.

Files

You should have the following files to use the program:

aeps.lisp

front-end.lisp

math.lisp

io.lisp

definitions.lisp

sets.lisp

methods.lisp

genss.lisp

select.lisp

process.lisp

cover.lisp

xp.lisp

compclass.lisp

An additional file called *compile.lisp* can be used to compile all of the necessary program files.

Suggestions

The code is written in standard Common LISP, so it can be compiled with no problems. Compiling the code is strongly recommended. If your compiler supports tail recursion elimination as an option, be sure to turn it on.

No session logging capability is built into the shell. For UNIX systems, I recommend running the program from within GNU Emacs, using the Inferior Lisp mode. The script utility of UNIX can also be used.

Reporting Bugs

All bug reports should be sent via e-mail to the author. Please describe the problem in detail and make sure that I can reproduce the erratic behaviour by sending a script of the session starting from the beginning.

Acknowledgements

I want to thank my advisor Henry Kyburg for his patience. The syntax of the language used in this implementation is heavily influenced by a similar program, CCRC, written by Ron Loui.

Sample Data File

```
;tweety first
(reset)
(set timing off)
(s% (fly bird) (100000 90000))
(s% (fly penguin) (1000 1))
(subs penguin (bird))
(mem tweety (bird))
(prob (fly tweety))
(mem tweety (penguin))
(prob (fly tweety))
(set confidence .99)  ;play with the confidence level
(prob (fly tweety))
(set confidence .51)
(prob (fly tweety))
(set trace off) ;trace demo
(prob (fly tweety))
(set timing on)         ;timing demo
(prob (fly tweety))

;interesting cases of dominance
(reset) ;clean start
(set timing off)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p c) (.15 .45))
(subs c (a))
(mem x (a b c))
(prob (p x))
(clear)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p c) (.35 .45))
(subs c (b))
(mem x ( a b c))
(prob (p x))
(clear)
(% (p a) (.1 .2))
(% (p b) (.3 .4))
(% (p (I a b)) (.35 .45))
(mem x (a b))
(prob (p x))

;show methods
```

```
(clear)
(% (property set1) (.1 .2))
(% (property set2) (.5 .54))
(% (property set3) (.1 .53))
(% (property set4) (.1 .6))
(subs set2 (set1))
(mem object (set1 set2 set3 set4))
(set trace on)
(set method 1)
(prob (property object))
(set method 2)
(prob (property object))
(set method 3)
(prob (property object))
(set method 4)
(prob (property object))
(set trace off)
(set method 1)
(prob (property object))
(set method 2)
(prob (property object))
(set method 3)
(prob (property object))
(set method 4)
(prob (property object))
```

Sample Session

A sample session using Sun Common Lisp and the sample data file:

```
> (load "mkuis")
;;; Loading binary file "mkuis.sbin"
;;; Loading binary file "definitions.sbin"
;;; Loading binary file "io.sbin"
;;; Loading binary file "math.sbin"
;;; Loading binary file "front-end.sbin"
;;; Loading binary file "process.sbin"
;;; Loading binary file "sets.sbin"
;;; Loading binary file "select.sbin"
;;; Loading binary file "methods.sbin"
;;; Loading binary file "genss.sbin"
;;; Loading binary file "compclass.sbin"
;;; Loading binary file "cover.sbin"
;;; Loading binary file "xp.sbin"

Mucit's Kyburgian Uncertain Inference Shell
Common Lisp Version 0.40  August 13, 1990


shell is reset

INF-SHELL==> (load "example")
LOADING ==>(RESET)

shell is reset
LOADING ==>(SET TIMING OFF)
LOADING ==>(S% (FLY BIRD) (100000 90000))
@ 0.95confidence level % (FLY BIRD) in [0.89817 , 0.90181]

LOADING ==>(S% (FLY PENGUIN) (1000 1))
@ 0.95confidence level % (FLY PENGUIN) in [0.00018 , 0.00548]

LOADING ==>(SUBS PENGUIN (BIRD))
LOADING ==>(MEM TWEETY (BIRD))
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
BIRD
```

```
All non-trivial candidates :
<BIRD , [0.89817 , 0.90181]>

And the reference class is :
<BIRD , [0.89817 , 0.90181]>
LOADING ==>(MEM TWEETY (PENGUIN))
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
<PENGUIN , [0.00018 , 0.00548]>
<BIRD , [0.89817 , 0.90181]>

conflict between
<PENGUIN , [0.00018 , 0.00548]>
<BIRD , [0.89817 , 0.90181]>

Winner by subset:
<PENGUIN , [0.00018 , 0.00548]>

Will combine the following
<PENGUIN , [0.00018 , 0.00548]>

And the reference class is :
<PENGUIN , [0.00018 , 0.00548]>
LOADING ==>(SET CONFIDENCE 0.99)
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
<PENGUIN , [0.00012 , 0.00840]>
<BIRD , [0.89754 , 0.90241]>
```

```
conflict between
<PENGUIN , [0.00012 , 0.00840]>
<BIRD , [0.89754 , 0.90241]>

Winner by subset:
<PENGUIN , [0.00012 , 0.00840]>

Will combine the following
<PENGUIN , [0.00012 , 0.00840]>

And the reference class is :
<PENGUIN , [0.00012 , 0.00840]>
LOADING ==>(SET CONFIDENCE 0.51)
LOADING ==>(PROB (FLY TWEETY))

Processing query :(PROB (FLY TWEETY))  Using Method 4
All supersets:
TWEETY
PENGUIN
BIRD


All non-trivial candidates :
<PENGUIN , [0.00060 , 0.00167]>
<BIRD , [0.89950 , 0.90049]>

conflict between
<PENGUIN , [0.00060 , 0.00167]>
<BIRD , [0.89950 , 0.90049]>

Winner by subset:
<PENGUIN , [0.00060 , 0.00167]>

Will combine the following
<PENGUIN , [0.00060 , 0.00167]>

And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>
LOADING ==>(SET TRACE OFF)

Processing query :(PROB (FLY TWEETY))  Using Method 4
And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>
```

```
Processing query :(PROB (FLY TWEETY))  Using Method 4
And the reference class is :
<PENGUIN , [0.00060 , 0.00167]>
Elapsed Real Time = 0.05 seconds
Total Run Time    = 0.06 seconds
User Run Time     = 0.05 seconds
System Run Time   = 0.01 seconds
Process Page Faults = 0
Dynamic Bytes Consed = 0

shell is reset
LOADING ==>(SET TIMING OFF)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P C) (0.15 0.45))
LOADING ==>(SUBS C (A))
LOADING ==>(MEM X (A B C))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
B
C
A


All non-trivial candidates :
<B , [0.30000 , 0.40000]>
<C , [0.15000 , 0.45000]>
<A , [0.10000 , 0.20000]>

conflict between
<B , [0.30000 , 0.40000]>
<A , [0.10000 , 0.20000]>

No reflection

conflict between
<C , [0.15000 , 0.45000]>
<A , [0.10000 , 0.20000]>

Winner by subset:
<C , [0.15000 , 0.45000]>
```

```
Will combine the following
<C , [0.15000 , 0.45000]>

And the reference class is :
<C , [0.15000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P C) (0.35 0.45))
LOADING ==>(SUBS C (B))
LOADING ==>(MEM X (A B C))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
A
C
B


All non-trivial candidates :
<A , [0.10000 , 0.20000]>
<C , [0.35000 , 0.45000]>
<B , [0.30000 , 0.40000]>

conflict between
<A , [0.10000 , 0.20000]>
<C , [0.35000 , 0.45000]>

No reflection

conflict between
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

No reflection

conflict between
<C , [0.35000 , 0.45000]>
<B , [0.30000 , 0.40000]>

Winner by subset:
<C , [0.35000 , 0.45000]>
```

```
Will combine the following
<C , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>

And the reference class is :
<(OR A C) , [0.10000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (P A) (0.1 0.2))
LOADING ==>(% (P B) (0.3 0.4))
LOADING ==>(% (P (I A B)) (0.35 0.45))
LOADING ==>(MEM X (A B))
LOADING ==>(PROB (P X))

Processing query :(PROB (P X))  Using Method 4
All supersets:
X
(I A B)
A
B


All non-trivial candidates :
<(I A B) , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

conflict between
<(I A B) , [0.35000 , 0.45000]>
<A , [0.10000 , 0.20000]>

Winner by subset:
<(I A B) , [0.35000 , 0.45000]>

conflict between
<(I A B) , [0.35000 , 0.45000]>
<B , [0.30000 , 0.40000]>

Winner by subset:
<(I A B) , [0.35000 , 0.45000]>

conflict between
<A , [0.10000 , 0.20000]>
<B , [0.30000 , 0.40000]>

No reflection
```

```
Will combine the following
<(I A B) , [0.35000 , 0.45000]>

And the reference class is :
<(I A B) , [0.35000 , 0.45000]>
LOADING ==>(CLEAR)
LOADING ==>(% (PROPERTY SET1) (0.1 0.2))
LOADING ==>(% (PROPERTY SET2) (0.5 0.54))
LOADING ==>(% (PROPERTY SET3) (0.1 0.53))
LOADING ==>(% (PROPERTY SET4) (0.1 0.6))
LOADING ==>(SUBS SET2 (SET1))
LOADING ==>(MEM OBJECT (SET1 SET2 SET3 SET4))
LOADING ==>(SET TRACE ON)
LOADING ==>(SET METHOD 1)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 1
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

4 is to combine
Creating 2 place XPs
5XP inference structures in this iteration
4 is to combine
Creating 3 place XPs
2XP inference structures in this iteration
XP generation finished.  A total of 7 inference structures generated
All candidates
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<(XP SET1 SET3) , [0.01220 , 0.21992]>
<(XP SET1 SET4) , [0.01220 , 0.27273]>
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>
```

```
<SET3 , [0.10000 , 0.53000]>
<(XP SET2 SET3) , [0.10000 , 0.56967]>
<SET4 , [0.10000 , 0.60000]>
<(XP SET2 SET4) , [0.10000 , 0.63780]>
<(XP SET3 SET4) , [0.01220 , 0.62846]>
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>

Considering :
<SET2 , [0.50000 , 0.54000]>

Disagreement with:
<SET1 , [0.10000 , 0.20000]>

Candidate survived
Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET1 SET4) , [0.01220 , 0.27273]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<SET3 , [0.10000 , 0.53000]>
```

```
Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Candidate survived
Disagreement with:
<(XP SET1 SET4) , [0.01220 , 0.27273]>

Doesn't dominate

Considering :
<SET4 , [0.10000 , 0.60000]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET2 SET4) , [0.10000 , 0.63780]>

Disagreement with:
<(XP SET1 SET3) , [0.01220 , 0.21992]>

Doesn't dominate

Considering :
<(XP SET3 SET4) , [0.01220 , 0.62846]>

Disagreement with:
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>

Doesn't dominate

Considering :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>
```

```
Disagreement with:
<(XP SET4 SET3 SET1) , [0.00137 , 0.29720]>

Candidate survived
And the reference class is :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>
LOADING ==>(SET METHOD 2)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 2
All supersets:
OBJECT
SET2
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

No reflection

After conflicts are resolved
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

3 is to combine
Creating 2 place XPs
3XP inference structures in this iteration
```

```
3 is to combine
Creating 3 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 4 inference structures generated
All candidates with XP's
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>
<(XP SET3 SET2) , [0.10000 , 0.56967]>
<SET4 , [0.10000 , 0.60000]>
<(XP SET4 SET2) , [0.10000 , 0.63780]>
<(XP SET4 SET3) , [0.01220 , 0.62846]>
<(XP SET2 SET3 SET4) , [0.01220 , 0.66507]>

Considering :
<SET2 , [0.50000 , 0.54000]>

Disagreement with:
<SET3 , [0.10000 , 0.53000]>

Doesn't dominate

Considering :
<SET3 , [0.10000 , 0.53000]>

Disagreement with:
<SET2 , [0.50000 , 0.54000]>

Doesn't dominate

Considering :
<(XP SET3 SET2) , [0.10000 , 0.56967]>

And the reference class is :
<(XP SET3 SET2) , [0.10000 , 0.56967]>
LOADING ==>(SET METHOD 3)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 3
All supersets:
OBJECT
SET2
SET1
SET3
SET4
```

```
All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

No reflection

After conflicts are resolved
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

Will combine the following
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

2 inference structures to combine
Creating 2 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 1 inference structures generated
Considering :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

And the reference class is :
<(XP SET2 SET3) , [0.10000 , 0.56967]>
LOADING ==>(SET METHOD 4)
LOADING ==>(PROB (PROPERTY OBJECT))

Processing query :(PROB (PROPERTY OBJECT))  Using Method 4
All supersets:
OBJECT
SET2
```

```
SET1
SET3
SET4


All non-trivial candidates :
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>
<SET3 , [0.10000 , 0.53000]>
<SET4 , [0.10000 , 0.60000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET1 , [0.10000 , 0.20000]>

Winner by subset:
<SET2 , [0.50000 , 0.54000]>

conflict between
<SET2 , [0.50000 , 0.54000]>
<SET3 , [0.10000 , 0.53000]>

No reflection

Will combine the following
<SET4 , [0.10000 , 0.60000]>
<SET3 , [0.10000 , 0.53000]>
<SET2 , [0.50000 , 0.54000]>

And the reference class is :
<(OR SET2 SET3) , [0.10000 , 0.54000]>
LOADING ==>(SET TRACE OFF)

Processing query :(PROB (PROPERTY OBJECT))  Using Method 1
4 is to combine
Creating 2 place XPs
5XP inference structures in this iteration
4 is to combine
Creating 3 place XPs
2XP inference structures in this iteration
XP generation finished.  A total of 7 inference structures generated
And the reference class is :
<(XP SET4 SET3 SET2) , [0.01220 , 0.66507]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 2
```

```
3 is to combine
Creating 2 place XPs
3XP inference structures in this iteration
3 is to combine
Creating 3 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 4 inference structures generated
And the reference class is :
<(XP SET3 SET2) , [0.10000 , 0.56967]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 3
2 inference structures to combine
Creating 2 place XPs
1XP inference structures in this iteration
XP generation finished.  A total of 1 inference structures generated
And the reference class is :
<(XP SET2 SET3) , [0.10000 , 0.56967]>

Processing query :(PROB (PROPERTY OBJECT))  Using Method 4
And the reference class is :
<(OR SET2 SET3) , [0.10000 , 0.54000]>
eof reached

INF-SHELL==> (bye)
Bye.
#P"/home/castor/u6/mucit/statinf/mkuis.sbin"
>
```

References

[1] Bülent Murtezaoğlu, Henry E. Kyburg. Think this up. Technical report, University of Rochester, forthcoming.

[2] Henry E. Kyburg. The reference class. *Philosophy of Science*, (50), 1983.

[3] Henry E. Kyburg. Evidential probability. Technical report, University of Rochester, forthcoming.

[4] Ronald P. Loui. *Man page for RC.*

[5] Ronald P. Loui. Computing reference classes. In *AAAI Uncertainty Workshop*, 1986.

[6] Ronald P. Loui. *Theory and Computation of Uncertain Inference and Decision.* PhD thesis, University of Rochester, 1987.