TBMLAB
draft

Philippe Smets
Brussels, Belgium
psmets@ulb.ac.be
http://iridia.ulb.ac.be/˜psmets

March 11, 2004

Contents

Chapter 1

Preamble

TBMLAB IS A FREEWARE. THE USER GUIDE (AND THE SOFTWARE) IS NOT AS POLISHED AS ONE CAN EXPECTED FROM A PROFESSIONAL WORK. USER WILL HAVE TO BE TOLERANT ABOUT ERRORS AND MISSING SECTIONS. BETTER A FREEWARE WITH SOME LIMITATIONS THAN A PROFESSIONALLY POLISHED BUT EXPENSIVE SOFTWARE. COMMENTS ARE WELOME. TO BE SEND TO THE AUTHOR EMAIL: psmets@ulb.ac.be

TBMLAB means TBM (for transferable belief model) LABoratory. It is a demonstrator for the TBM, a model for the representation of quantified opinions, based on what Shafer has described in his 1976 book, to which many new concepts, clarifications and results have been added.

This software is a freeware developed with Thierry Denoeux (Université Technologique de Compiègne). Initial developments were done in collaboration with Prakash Shenoy and Phan Hong Giang (University of Kansas) with a grant from Raytheon. Helps from Pat Cappelaere, Ha Duong Minh, Christophe Marsala, Pierre-Henri Wuillemin, Branko Ristic are highly appreciated.

This software can be used freely but under the full responsibility of the user and in accordance with the copyright statement. The authors cannot be held responsible for any problems resulting from the use of this software. It may not be sold. Any modification must be clearly made explicit to users.

Formally the TBMLAB is a demonstrator for the TBM, but in practice it can be seen as a demonstrator for belief function theory. It covers essentially:

- the probability based belief function theory
    - the theory developed by Dempster and by Shafer
    - the theory of hints of Kohlas and Monney
    - the probabilistic argumentation system of Haenni, Kohlas and Lehmann.
- the non-probabilistic belief function theory
    - the theory presented in Shafer's 1976 book
    - the transferable belief model (TBM) of Smets

It does not cover upper and lower or imprecise probability theory. The term Dempster-Shafer theory is not used as it is unclear which of the above models it represents. The use of the ‘Dempster-Shafer theory’ label can only lead to a confusion between imprecise probability theory, probability based belief function theory and non-probabilistic belief function theory, these models sharing some common properties, but also serious differences.

Given the author past work, the TBMLAB is of course developed with the TBM as the background model, but many tools are applicable or adaptable to other models among those listed above.

Chapter 7 introduces the TBM. It is useful to fix the definitions of the terms used in the TBMLAB user guide, but can be skipped by those familiar with the TBM.

Chapter 2

Presentation

2.1 Introducing TBMLAB

TBMLAB is a MATLAB program developed under MATLAB 5.2, Mac0S Classic. In order to be easily portable, it does not use programs and functions. written in other languages. It runs also under Window (PCWIN), MacOSX with MATLAB 6.5. For Unix, it has not yet be really tested, but may work.

Data for TBMLAB can be introduced

- **on-line** through many GUI's or
- in **batch mode** where instructions are loaded from a .txt file.

The user can:

1. use TBMLAB to **compute** belief potentials using some combinations of already existing belief potentials. Results can be stored and saved.
2. ask TBMLAB to **display details** of belief potentials.
3. introduce an **evidential network**, and ask TBMLAB to propagate belief potentials in the network. Two kinds of networks can be created:
   - an evidential network where the belief potentials are defined on product spaces of variables, and
   - a conditional evidential network where the belief potentials are conditional potentials defined on one space given the singletons of a second space.
4. run **tutorials**.
5. run the **TBMLAB Tabulator**, a tabulator specially oriented toward belief function computations.
6. use **specialized modules** centered on particular applications.

Sections of this User Guide cover:

- the installation instructions (chapter 3).

- a set of demos on how to use TBMLAB. Should be read and run by beginners (chapter 4).
- the batch-mode syntax where we explain how to write instructions for using TBMLAB in batch-mode (chapter 5).
- the reference manual (chapter 6).
  - the nature of the objects (chapter (section 6.1).
  - the details about the menus (sections 6.2 to 6.7).
  - the use of the belief potential editor (section 6.8).
- a summary of the theory of the TBM (chapter 7).

2.2 The purpose of TBMLAB

TBMLAB has been built in order future users of the TBM can learn its behavior and its capacities. The authors realize that the users of belief functions have often difficulties in understanding the model. Theoretical papers are numerous, but we feel that a demonstrator could help a lot in understanding the model and its capacities.

It is not a toolbox. It is not a set of functions that can be used in another programs, even if it could be done in some cases. The MATLAB codes are not documented for a possible use outside TBMLAB.

2.3 The overall organization of the TBMLAB

TBMLAB is a self contained program. It is made of an initial window (the **Root Window**) to which several **menus and submenus** are attached. By selecting a menu, the user is sent into a sequence of subprograms that performs the selected task. These tasks will have their own windows, menus and button lists. In case of normal use, the last step consists in quitting the task, in which case the user comes back to the initial Root Window, and can select a new task.

Useless menus and buttons are normally dimmed or invisible.

Contextual helps are available at many places and on many buttons. They can be displayed by selecting the appropriate menus.

The architecture of TBMLAB is based on **objects**.

2.4 The major objects

2.4.1 Example

To understand the objects, consider the next example.

Suppose You, the belief holder, want to enter Your beliefs about the yearly income of Aude, Bart, Carr, Dany during years 2001, 2002 and 2003. The income can be expressed in Euro or as ‘high’,‘medium’, ‘low’.

There are thus 12 variables on which beliefs can be defined:

Income(Aude,2001), Income(Aude,2002), Income(Aude,2003),
Income(Bart,2001), Income(Bart,2002), Income(Bart,2003),
Income(Carr,2001), Income(Carr,2002), Income(Carr,2003),
Income(Dany,2001), Income(Dany,2002), Income(Dany,2003).

Frame. We create the next four frames, each one storing the list of possible values that can be taken by a property:

- a frame Fpeople with four elements: Aude, Bart, Carr, Dany.
- a frame Fyear with three elements: 2001, 2002, 2003.
- a frame FincE with elements are the integers from 0 to 500.000. It covers the income in Euro.
- a frame Finc3 with three elements called: high, medium, low.

Structure. As far as FincE and Finc3 concern the same property expressed on different scales, we create a structure SInc that expresses the concept of income, and which is made of two frames FincE a nd Finc3. For completeness we also create the structures Speople and Syear, both with one frame, Fpeople and Fyear, respectively.

Mapping. The relation between the two frames of structure Sinc is stored in an object called mapping that states that Low = [0,29999], Medium = [30000,99999], High = [100000,500000].

Attribute. The 12 variables listed above are based on one attribute Ainc expressed on the structure Sinc and which has two indexes Iperson and Iyear, which values are the elements of the frames Fpeople and Fyear, respectively.

Variable. Finally, consider the variable Income(Carr,2002). It is based on the attribute Ainc where the indexes have been instantiated as people = Carr and year = 2002. Furthermore let Your beliefs about this particular variable be expressed on the frame Finc3. Similar construction must be performed for every variable.

Belief. All needed object are constructed. Belief can be allocated to any subsets of these 12 variables. Beliefs can be univariate when defined on the space made of only one variable, or multivariate when defined on the product space made of several of the 12 variables.

2.4.2 The objects

More formally, the objects are:

- Frame: the list of possible values that can be taken by a variable or an index.
- Structure: a set of frames that concern the same variables.

- Mapping: the relation between the frames that belong to the same structure.
- Attribute: a property that defines a variable. It can be indexed.
    - Index: that defines whom or what the attribute is attached to.
- Variable: the object of the beliefs, which value is not known but for which the user has some opinions quantified by the belief function. It corresponds to an attribute (with instantiated indexes when attribute has indexes) defined on a given frame.
- Belief: the weights of the opinions about the actual value of the variable. It has its own sub-objects.
    - Belief Potential: one belief potential (bp) for each value of the conditioning variables. It has its own sub-objects.
        - Form: weighted opinions can be represented under several functions which are usually in one to one correspondence, but enhance different properties of the beliefs.

The apparent complexity of the objects reflects only the large variety of applications to which the TBM can be applied.

2.5 TBM tasks under TBMLAB

TBMLAB can perform the next TBM tasks. Tasks 1 to 10 are found in the Tools menu, whereas the last ones are accessible through the Edit/Belief menu, in the potential editor window.

1. **VBS.** You select several bba's and several variables. The Valuation Base System algorithm (VBS) of Shafer-Shenoy propagates the beliefs and projects them on the selected variables. Entered beliefs can be both joint beliefs or conditional beliefs.
2. **Graphical Method.** You run the VBS under graphical mode. You get a display of the Evidential Network (EVN). or of the Conditional Evidential Network (CEVN). Beliefs are propagated using the VBS. You can graphically display the pignistic probability function for every variable.
3. **Distinct Combination.** You select a set of bba's. They are combined by a mixture of negations, conjunctions and disjunctions using the negation rule, the conjunctive combination rule or/and the disjunctive combination rule. Results are produced in their joint space. This brute force combination operation is not as efficient as the VBS algorithm. It works for small frames.
4. **Cautious Combination.** You select a set of bba's. They are combined by the cautious conjunctive combination rule which is the commutative, associative and idempotent rule of combination.

5. **GBT: X/T both discrete.** The General Bayesian Theorem (GBT) requires a belief object with conditional potentials on X for every $t_i \in T$. You can enter an a priori belief on T and an a priori belief on X (which corresponds to a doubtful, noisy observation). You can run either the GBT to get the resulting belief on the T space or the DRC to get the resulting belief on the X space (forward or backward propagation). Both X and T must be discrete variables (not interval variables like those based on Integers and Continuous frames (see 6.1.3))

6. **GBT: X Continuous-Integers, T Discrete.** This form of General Bayesian Theorem (GBT) requires a belief object with conditional potentials given as a probability density function (pdf) on a Continuous or Integers variable X, and this for every $t_i \in T$. X must be unidimensional. This pdf can have two meanings.

   - The pdf represents a Bayesian belief function (bbf) on X. Then the pdf can be one of {binomial, beta, gamma, gaussian, laplace}.
   - The pdf is the pignistic transformation of some unknown underlying bba on X (case Betf). In that case we compute the q-least committed bba on X isopignistic with Betf (isopignistic means that its pignistic transformation is equal to the entered pdf).

   A priori on T is vacuous and a priori on X is categorical (one interval of X receives a mass 1). The computation produces the a posteriori on T.

7. **TBM Tabulator.**

   A tabulator designed to run TBM operations when cardinalities are small (3 to 5).

8. **Get New Forms.** Given a potential, computed any of {m, b, bel, pl, q, wc, wd, M, Bel, Pl, Q, BetP} functions.

9. **Extension/Marginalization.** Given a potential defined on space X, compute its projection on space Y. It performs both marginalization and vacuous extension.

10. **EvClus: dissimil → m.**

    Input is a dissimilarity $N \times N$ matrix $D = [d(i,j)]$ where $d(i,j)$ is the dissimilarity between object i and object j.

    It is assumed :

    - individual can be categorized into N classes (an input parameter),
    - for each individual, there is a belief potential (to be determined) representing our knowledge about the class to which it belongs,
    - for every pair (i,j) of individuals, let $pl\{i,j\}(S)$ be the plausibility that they belong to the same class (computed from last belief potentials),
    - the condition: $d(i,j) > d(k,l) \rightarrow pl\{i,j\}(S) < pl\{k,l\}(S)$.

    The EvClus (evidential clustering) algorithm determines the belief potentials that satisfy (as well as possible) these conditions. Based on Denoeux and Masson's work.

11. **LCP BetP Pl** Given the values of BetP or of pl on the singletons of a variable, TBMLAB determines the q-least committed isopignisitc (BetP case) or isoshadow (pl case) bba.

12. **Best supported set.**

    Given a bba on a large frame, determine the smallest subset with the largest weighted plausibility. This permits to reduce greatly the set of elements in which the truth belongs. Based on Appriou's work.

Chapter 3

Installation

3.1 System requirements

MATLAB 5.2 or higher.

TBMLAB has been tested

- on Mac, under OSX-Classic, MATLAB 5.2, and under OSX, MATLAB 6.5 (possible bugs).
- on PC, use MATLAB 5.2 or more recent versions. It works nicely with Windows2000. With Windows95, user may miss red backgrounds in the user interface and scrolling menus attached to fields.
- on Unix stations, with MATLAB 6.5. Seems working but the numerous machine dialects make it hard to test it on every stations, and there seems to be some pending problems.

The screen size is supposedly 1024 x 768. On smaller screen, GUI's get messy. We do not profit from larger screens as GUI's become inconsistent when moving from one screen to another.

3.2 Installation procedure

Follow the next instructions if this is a first installation. For upgrade, see section 3.4.

1. Create a folder called TBMLAB_Folder. It can be located anywhere.
2. Unzip the file TBMLAB_x.y.zip, where x.y is the version level.
3. Drop the unzipped folder TBMLAB_x.y in the folder TBMLAB_Folder you have just created.

The paths to the files of TBMLAB_x.y are fixed inside the folder TBMLAB_x.y. So don't rearrange the folders within TBMLAB_x.y, else tbmlab could crash.

3.3 Running TBMLAB

1. The first thing you must do is creating the path to the TBMLAB files so MATLAB knows where files are located. It can be done in two ways.
   (a) If the Add subfolders option is available on your MATLAB Path, use MATLAB Path and Append or Insert all the files using the Add subfolders option. Save the result. This method is supposedly the best one. If the option Add subfolders is not available use the next options.
   (b) Create one path to the folder TBMLAB_x.y using the MATLAB Path function. Then in the MATLAB Command Window, type tbmlab and hit the RETURN key. TBMLAB creates all the paths and return to MATLAB. Paths are not saved, so maybe do it using your MATLAB Path function.
2. Once paths have been created, type tbmlab in MATLAB Command Window and hit RETURN key: you enter in TBMLAB program.

Note: for debugging needs, use tbmlab(0) and you get a better trace of the error, but the windows do not close nicely. To close them, go to the MATLAB Command Window, hit Control_C twice, and run the instruction 'close all force' in order to close all the windows created by TBMLAB.

3.4 Updating TBMLAB with a new version

To install a new version, delete all paths leading to any TBMLAB folder (ESSENTIAL as MATLAB uses without problem trashed folders), then throw the old version by dropping the folder TBMLAB in the trash.

If your MATLAB asks for a file-by-file path deletion, run delete_path. In theory, it deletes automatically all the paths related to TBMLAB_x.y, but be patient, it can be slow. Besides it might not work, as it seems MATLAB Path handling varies from machine to machine. Sorry about it. The trouble is that trashing folders is NOT sufficient as Paths are not deleted.

Personal data. The only files that can contain your data are in folder TBMLAB/User_Data:

1. Application_Data: contains a set of files, one for each application. In each application subfolders, one could find
   - a batch_mode text file which name contains .batch, used to run data in batch mode.
   - a binary file, ending with .obj.mat. It contains all the data relative to the objects frame, structure, mapping, attribute, variable and belief, saved by TBMLAB when you ask for it.
   - a binary file, ending with .mat and with .EVN., that contains the data relative to the EVN saved by TBMLAB when you ask for it.

  - other files saved during the run relative to this application, as with the Print Report, or View options, or the batch mode REPORT option.

2. Batch_Mode_Matrices: it contains all possible instructions that can be used to run beliefs in batch mode. Useful for creating a batch mode files, using lot of copy paste.

3. Other_Data: data used by special applications. Avoid using it and prefer putting all data for one application in one folder in Application_Data folder.

4. Preferences_Data: it contains the files with the default and the personal preferences used by TBMLAB.

5. Printed_data: data saved with a print option, like the .eps files with your saved plots.

In these folders you can replace the new ones with your old ones if you want it, or drop your own subfolders in the corresponding new folders. For instance if you had a folder with data you created and called my_data located in Application_Data, move it into the new Application_Data folder before trashing your old TBMLAB.

Then reinstall the paths (see section 3.3) and run TBMLAB (see section 3.4.1).

Winzip. For PC, it seems that 'winzip with overwrite' does not behave the way it should. Don't use it.

3.4.1 Launching TBMLAB

Once you have typed tbmlab, you get a 'nice' front page. It is nice on my Mac, but apparently not under some Unix. Sorry for the Unix people. Just click anywhere to proceed.

You receive then two windows, called the Root Window (located at left) and the General Window (located at right).

- The Root Window. It provides some helps concerning the problem you are facing. It permits access to the primary menus. Its Window Name tells if the pull down menus are enabled (ON) or disabled (OFF)..

- The General Window. It is used for various purposes depending on the task under process.

In the **Root Window**, you receive a **welcome message**. **Menus** are enabled. Your task will be to navigate among these menus. So read at least once what each menu does, even if it will be unclear for many of them.

Some hints. To decide what to do next, a good rule is 'FOLLOW the RED'. It means that what you have to do is usually in red, like a red button, or the red title of a list, or a red button list. Look for the red, and often you will know where you should transfer your pointer for clicking. The green color means 'just go on'.

When stuck, type **CTRL+C**. It will make you quit TBMLAB and bring you back to MATLAB Command Window. You may loose some data (those recently changed and not yet saved. Sorry.) If the windows are still present, type CTRL+C and maybe again type CTRL+C. If the system does not obey to the CTRL+C, move the windows so that you can put your pointer on the MATLAB Command Window and try CTRL+C again once or twice. Normally it works, else sorry, use a forced quit to leave MATLAB.

If your CTRL+C has worked and the TBMLAB windows are still open, go to the MATLAB Command Window and type the instruction '**close all force**', all TBMLAB windows will be closed, and your machine is normally clean (except that some MATLAB GUI parameters have been changed and MATLAB does not restore their initial default values).

The delivered TBMLAB comes with **max help level** option. It is assumed users are beginners, and need as much help as possible. See section 6.7 on how to change the help level.

3.5 On Line Execution

User gets GUI's and navigates among them, selecting menus and buttons and answering queries. Learning a special programming language in not needed.

3.6 Batch Mode Execution

TBMLAB can be driven in batch mode. The Batch mode is not as powerful as the on_line mode, but it is sufficient for most practical problems. User must build a text file (.txt for instance) that lists all the instructions according to syntax detailed in chapter 5.

With Word, use the Save as with the option Text Only with Line Breaks. With Mac TextEdit, in menu TextEdit/Preference, panel, New Document Attributes, Select : Plain Text.

3.7 Some comments.

- MATLAB behavior is not always obvious for the user. When you are asked for **selecting a button in a panel**, a RETURN works sometimes, in which case it applies the reddest button option (always at left). If TBMLAB does not react, select the panel (click anywhere in the panel), and then select the button (click on it). For non-Mac users, if it still does not work, try to use the right button of your mouse.

- For **multiple selections** in lists, you must use Capital-left mouse button, i.e. use the mouse whole pushing on the Capital key.

- It is advised that beginners use the **help level** as max. If it is not yet so, select the menu Tools/Preferences/Help Level and in the panel that pops up, select the level max, so you get all possible helps.

- **Menus** are attached to windows in Window, Unix and Mac-OSX (Window, Unix and Mac-OSX versions use the same presentation), or are located in top bar for Mac OS9).
- Beware: you must quit TBMLAB when you **move from one problem to another**. It is required in order to properly initialize all variables. It is not required to quit Matlab.

Chapter 4

Using TBMLAB: Demos

4.1 Introduction

We detail both the batch-mode and the on-line creation of several illustrative examples. In the folders of the User_Data/Application_Data folder, we usually provide

1. a .batch.txt file with the instructions used to create the data for the given application.
2. a .obj.mat file created for the given application. It contains the binary data saved while executing the batch_mode files or upon user's request.

To study the demos, a good strategy consists in

1. reading the batch-mode instructions,
2. running the on-line mode instructions to see how it works,
3. loading the saved binary data when re-running the example during a later run.

4.2 A problem with logical propositions

In MATLAB Command Window, type tmblab RETURN. You get a welcome display. Click anywhere on it. You enter in the Root Window, located at left. Wait for the green button. Then all actions are performed using the menus and the special panels that will pop up.

Maybe the paths had not yet been saved. Then TBMLAB creates them and quits itself. A good idea is to save your paths now, but it is not necessary. Once you are ready, rerun tbmlab.

4.2.1 The problem

The first very simple problem concerns the rule 'if it rains at my office, then the ground is wet at home', which you believe to be true with degree .7. You also have some belief (.8) that it rains at your office. What is your belief that the ground at your home is wet?

4.2.2 Batch-Mode

General rules. Comments start with a % symbol. Empty lines or lines with only spaces are skipped. xSHN, xFLN denote the x short name and x long name where x can be F, S, A, V or B for frame, structure,attribute, variable, belief, respectively. If instructions are not clear, don't stop and proceed. Things will (hopefully) become clear later.

```
TBMLAB BEGIN          % instructions start here

FRAME                 % creating a Logical frame
FSHN = FT             % its short name is FT
FFLN = Logical        % its full name is Logical
FTYP = Logical        % type for frame is logical
FCRD = 2              % cardinality is 2 (default is F,T

STRUCTURE             % creating a structure
SSHN = SFT
SFLN = Logical
SMRF = FT             % based on frame FT

ATTRIBUTE             % creating an attribute
ASHN = r
AFLN = it_rains
ASTR = SFT            % based on SFT structure

ATTRIBUTE             % creating an attribute
ASHN = w
AFLN = street_is_wet
ASTR = SFT            % based on SFT structure

VARIABLE              % creating a variable
VSHN = r
VFLN = it_rains
VATT = r              % based on r attribute
VFRM = FT             % based on FT frame

VARIABLE              % creating a variable
VSHN = w
VFLN = street_is_wet
VATT = w              % based on w attribute
VFRM = FT             % based on FT frame

BELIEF                % creating a belief
BSHN = r_0
BFLN = rain
BAGN = You            % the agent, the belief holder
BTME = 1              % the time when beliefs are held
BDVR = r              % the domain variable short name is r
POTENTIAL             % announcing a new potential
FORM                  % announcing a new form for this potential
```

```
PFRM = m                % the form is a m function
PREP = selected         % based on selected focal sets
PBEG                    % begin a the mass creation
.8 D 2                  % m=.8 given to focal set decimal = 2 (T)
PEND                    % end of mass creation
                        % unallocated mass given to universe

BELIEF                  % creating a belief
BSHN = r>w
BFLN = rain->wet
BDVR = r w
POTENTIAL
FORM
PFRM = m
PBEG
.7 L r impl w           % m=.7 given to set of worlds where r implies w
PEND

COMPUTATION             % computation starts here

VBS                     % apply VBS to propagate beliefs
BSHN = post_w           % store results in belief short name post_w
BFLN = posterior_wet    % store results in belief full name post_w
BELF = r_0,r>w          % use these two belief objects
PVAR = w                % marginalize on the variable 'wet'

TBMLAB END              % end of all tasks
```

You could run this file, but you would not learn much out of it. So instead, we proceed now with the on-line mode in order to create the data and run the VBS.

4.2.3 Phase 1 : Variables

You must create two variables which are logical propositions. The first proposition is 'it rains' and the second 'ground is wet'.

Select the menu Edit/Short Cuts/Logical propositions. In Get Propositions window (panel at right), a small panel pops up and you are asked for the number of propositions you want to create. Select (click in) the answer area (the white area where you see 0) replace the 0 by a 2 (just overtype) and click on OK.

Another panel pops up. It asks for a short name for each proposition. Default is p1 and p2. Replace p1 by 'r' and p2 by 'w'. Click OK. Then a similar panel asks for long names and proposes as default the short names. Change them into 'it_rains' and 'ground_is_wet'. Click OK.

The two variables have been created (and even a logical frame, a structure and an attribute have been automatically created, but you can just as well forget about it).

You are back in the Root Window (the one with the green button).

4.2.4 Phase 2 : Beliefs

You must now create the two belief functions that concern these variables. You select the menu Edit/Belief. At right appears the Belief Editor window. Look at its own menus. In the File menu, you will see the quit menu that permits you to leave the Belief Editor and go back to the Root Window. In the Help menu (called TBM Help for Mac OS9 as OS9 refuses to abandon its own Help that do not interest us here), you get two submenus: Window Helps On and Button Helps On. Select Window Helps On and you get a help in the Root Window (at left). Look again in the menu Help, you have now Window Helps Off. If you select it, the help text disappears, and the menu returns to Window Helps On. So you ask and cancel the window helps at will.

The Button Helps are helps attached top the buttons themselves. When you select the menu Button Helps On, and when your pointer pass over a button, a help related to the button itself pops up in the left window. The menu has become Button Helps Off. If you select it, Button Helps disappear. So you ask and cancel the buttons helps at will.

Put both Helps on, the helps may help you.

The Window Help explains you what are the objects used by TBMLAB and what are the fields of the belief object (provided you did not choose Min help level). Understanding this structure is not really necessary for a casual user.

In the Belief Editor window, you get a button list with title BELIEF, the list of the already available variables (your two propositions) and the list of the already available beliefs (empty as there are none yet).

Select the button new.

You get a query dialog panel in the Belief Editor window. It is made of queries and fields ready for the answers. It proposes default values for most queries. You modify them by just typing over the displayed texts. Each query corresponds to one field from the belief object where the belief function is stored. The short name and full name are just up to you. We have to store two belief functions. Their short names could be 'rw' for the implication 'rains implies wet', and 'r0' for the prior belief on 'it rains'. Full names could be r->w and rain_prior.

Let us enter the implication first. We type rw and r->w for the short and full names, respectively. Disposable is left N (for No, it means that the belief may not be disposed off (thrown away) when TBMLAB needs some memory space). Agent, time and evidential corpus can be left blank. Click OK to say your task is over and TBMLAB must proceed.

It asks what are the variables involved in that belief function. You select them in the Variable list using the key for multiple selection (Cmd Click or Capital Click on Mac, Capital Click for PC). Select both listed variables, then DONE. The belief object has been created and appears in the Belief list.

The next question just asks you if you want to enter some values for the belief function. Click Yes. You get a Belief Potential window at right. In the Root Window you get a new help text explaining what are the belief potential objects. In the Belief Potential window, you get several lists of possible actions, each in a button.

The four lists of buttons at left concern potential creation, the upper right proposes display of potentials, the lower right is control oriented.

Button Help menu work as in the previous window. It should be on. If not, select it. There is no Window Help menu as it is always present (a choice made by the author).

You want to create a bba so you choose in the upper left panel the option Logical expression (you want 'rains implies wet', thus an implication, thus a logical expression).

Still anther window, the 'Bba Editor: logical mode' window. Four panels pop up, one with the list of all involved variables, one with a list of legal logical operators, one with control instructions, and below one empty panel where the logical expression will show up as you build it. You want r implies w, so you select successively the buttons r, then impl, then w, then OK to end the expression. You should have noticed that the visible options change all the time. In fact you get only those operators and variables that you may use at the next selection. We hope this will avoid erroneous manipulations when using of the expression builder. Each option (button) will always reappear at the same location, but they can be visible or invisible according to the last selection. For instance behind a 'not' you may not enter an 'and' operator, behind a variable you may not enter a 'not' operator, etc...

When you select OK, an empty box at lower right pops up, and is ready to receive the mass to be given to the implication. Enter .7, then click OK.

TBMLAB is ready and asks for the next expression. There is no need to enter any other expression. So select the menu File/Quit to return to the Belief Potential window. The same action could be achieved by selecting the quit button in the control panel (under the OK button). TBMLAB allocates the unallocated mass (.3) to the universe. The first bba has been created. You leave the Belief Potential window by selecting the menu File/Quit or the Quit button in panel at lower right. You are back into the Belief Editor window.

Let us create the second belief function. In the button panel, several options appear now that you don't need but they were forbidden previously as non-applicable (like displaying, deleting or saving a belief object when none exists).

You select again New. In the query panel, you enter the short and full names 'r0' and 'rain_prior'. Click on Done. For the next question, select the 'it rains' variable. Then select Done, For the 'Go to potentials', select Yes. To create the potential, this time use the option 'Sel.elem: No Cont/Int'. It is simpler. You get the 'Bba Editor: set mode' window. It displays a list of every element of the variable space on which your belief function is going to be built. Here there is just one variable (r) and two possible values (T for true and F for false). A set is defined as one column. At top, the number (1.0 here) is the mass given to the focal set described in the column. Below, there is an index, and still below the focal set. The focal set if the set of the elements indicated by a x. By default a mass 1 is given to the universe (F,T) as both the F and T lines have a x. Select the cross next to F, it disappears. The focal set is now T. Select the mass by clicking on it. In the box that shows up, enter .8, RETURN. The .8 basic belief mass given to the focal set T. You could proceed for the .2 mass, but as previously said TBMLAB allocates the unallocated mass (.2) to the universe. So just select the menu File/Quit, and your second belief potential has been built. Select menu File/Quit twice and you are back to the Root Window (with the green button) which is the root of TBMLAB.

Bravo: all beliefs have been built. We proceed now with the computation.

4.2.5 Phase 3 : Computing

You want to compute what is the belief that the ground is wet at home. Select menu Tools/Valuation Base Systems. At right you receive the Run VBS Window with the list of available belief objects. Multi-select both beliefs in the belief list. Select the OK button. You get a list of available variables. Select the variable on which you want to marginalize the results, the second here (street_is_wet). Select OK. Computation is performed by TBMLAB.

You get the Display of belief window. Its name tells you the name of the built belief object, and its potential number. The window presents the results. It shows:

- at lower left, a plot with the most probable elements presenting Bel, BetP and Pl for each of them (Bel and Pl are the normalized values of bel and pl), BetP is indicated by the dark red line, and the upper and lower limits of the box in pale red are the Pl and the Bel values. One column corresponds to one singleton and is referred by a number.

- at upper left, the meaning of the just mentioned numbers are given here. The table presents the numbers used for the plot, (N = 1, 2), the values of BetP (.78 and .22, see the dark red lines in the plot) and the meaning of the numbers (1 for wet = True, 2 for wet = False). So the most probable answers is T with a pignistic probability of .78

- at right, you get a list of the largest basic belief masses and their focal sets. Some comments precede the 'Mass given to the empty set' which is 0 here. You get then the uncertainty measures for the normalized and the unnormalized bba (equal here as $m(\emptyset) = 0$).

  - the non specificity : $\sum_A |A| m(A)$,
  - the cardinality : $\sum_A \log(A) m(A)$.

  Then come the focal sets and their masses. The largest mass is .56 given to the focal set 'w=T'. Then the mass .44 is given to the foal set 'w=T or w=F' thus the universe.

Select the menu Tools/Variable names. You get a small window with the list of short and full names of every variable involved in the plot, a nice tool when there are lots of variables and you don't remember their short names. To close the window use the window close button (the X at right for Window and the □ at left for Mac).

With menu File/Quit, you are back to the Root Window with all its pull_down menus.

To quit the session, select menu File/Quit, and confirm by selecting the Yes.

Bravo, you finish your first experience with TBMLAB.

4.2.6 Comments

You should notice that most sessions with TBMLAB proceed through three phases:

1. building the variables on which the beliefs are defined,

2. building the belief potentials, usually the basic belief assignments,
3. computing new beliefs form the previously created ones.

We will thus follow that schema in the next examples.

4.3 FTA Diagnosis problem

4.3.1 The problem

The purpose of the next example is to learn the General Bayesian Theorem and directed and undirected evidential networks. The example is medically not serious, it is given as an illustration.

We consider three mutually exclusive diseases : appendicitis, colic and infarction.

There are three symptoms: fever, abdominal pain and thoracic pain.

For each possible disease, we have a belief about the status of each symptom.

For a given patient, we collect information about the symptoms, which induce some belief about the symptom status.

There are two ways to solve this problem:

- using conditional beliefs. The knowledge about the symptoms given the diagnosis is described by conditional beliefs $bel^{symptom}[disease]$. This is the TBM analogous of the conditional probability functions. The resulting network is similar to the DAG described in probability theory.
- using joint beliefs. The knowledge about the symptoms given the diagnosis is described by a belief function defined on the $symptom \times disease$ space. In practice, usually, each knowledge can be described as a set of weighted implications like ‘Infarction implies Fever is none’ (weight .6) and ‘Infarction implies Fever is none or low’ (weight .3).

We consider the two methods in parallel.

4.3.2 Batch-Mode

The bath mode data are in the application folder DiagFTA.

1. file DiagFTA_CBf.batch.txt: based on conditional beliefs (see left column)
2. file DiagFTA_JBf.batch.txt: based on joint beliefs (see right column)
3. file DiagFTA_GBT.batch.txt: based on conditional beliefs, but using the GBT program itself (see section 4.5)

I do not repeat all the comments of the previous example. Just a few new instructions deserve comments.

FINISH_VAR. Suppose I want to create a variable with short name shn (VSHN = shn) , with full name fulln (VFLN = fulln) (same label as short name if omitted) and announces that its frame is frameX (VFRM = frameX) already created. With the instruction FINISH_VAR at the end of a VARIABLE instruction set, TBMLAB creates

- a structure with short name Sshn, full name Sfulln, and a most refined frame frameX,
- an attribute with short name Ashn, full name Afulln, and with structure Sshn.

This all what we need in such simple cases and we can avoid the related STRUCTURE and ATTRIBUTE instruction sets.

For the conditional beliefs, BCVR is the short name of the conditioning variable, BPCN is the value of an element of the conditioning variable. So if BDVR = $VarX$, BCVR = $VarT$ which frame elements are $t1, t2, t3$ and BPCN = $t2$, the created bba corresponds to $m^{VarX}[t2]$. The other instructions have already been introduced. Missing conditional potentials are vacuous. The order under which the conditional beliefs are entered in the BELIEF instruction set is irrelevant. Do not enter the same conditioning element twice (it is stupid, and might crash TBMLAB).

We use the belief object DiagUni (last belief object at right) in order to mimic exactly the conditional case. It just declares that the three diagnoses are mutually exclusive and exhaustive.

The use of conditional beliefs is not only more natural, it is also more efficient as seen by the length of the instruction sets.

```
TBMLAB BEGIN                        TBMLAB BEGIN
% The conditional belief approach   % The joint belief approach

% Example with 3 diagnoses: Diag = Appen,Colic,Infar
% and 3 symptoms: Fever (nlh), Abd_pain (nms), Thor_pain (nms)
% where nlh and nms denote their possible values with
% nlh = none,low,high,
% nms = none,mild,strong

FRAME                               FRAME
FSHN = nlh                          FSHN = nlh
FTYP = Classes_Ordered              FTYP = Classes_Ordered
FCRD = 3                            FCRD = 3
ELEM = none,low,high                ELEM = none,low,high

FRAME                               FRAME
FSHN = nms                          FSHN = nms
FTYP = Classes_Ordered              FTYP = Classes_Ordered
FCRD = 3                            FCRD = 3
ELEM = none,mild,strong             ELEM = none,mild,strong

FRAME                               FRAME
FSHN = diag                         FSHN = yn
FTYP = Classes_Un_ordered           FTYP = Binary
FCRD = 3                            FCRD = 2
ELEM = Appen,Colic,Infar            ELEM = yes,no

                                    FRAME
                                    FSHN = diag
```

```
                              FTYP = Classes_Un_ordered
                              FCRD = 3
                              ELEM = Appen,Colic,Infar

VARIABLE                      VARIABLE
VSHN = Fever                  VSHN = Fever
VFRM = nlh                    VFRM = nlh
FINISH_VAR                    FINISH_VAR

VARIABLE                      VARIABLE
VSHN = Abd_pain               VSHN = Abd_pain
VFRM = nms                    VFRM = nms
FINISH_VAR                    FINISH_VAR

VARIABLE                      VARIABLE
VSHN = Thor_pain              VSHN = Thor_pain
VFRM = nms                    VFRM = nms
FINISH_VAR                    FINISH_VAR

VARIABLE                      VARIABLE
VSHN = Diag                   VSHN = App
VFRM = diag                   VFRM = yn
FINISH_VAR                    FINISH_VAR

                              VARIABLE
                              VSHN = Colic
                              VFRM = yn
                              FINISH_VAR

                              VARIABLE
                              VSHN = Infar
                              VFRM = yn
                              FINISH_VAR

                              VARIABLE
                              VSHN = Diag
                              VFRM = diag
                              FINISH_VAR

% rules on fever              % rules on fever

BELIEF                        BELIEF
BSHN = Fev_D                  BSHN = RAF
BFLN = Fever_Given_Diag       BFLN = RAppendFever
BDVR = Fever                  BDVR = Fever App
BCVR = Diag                   POTENTIAL
POTENTIAL                     FORM
BPCN = Appen                  PFRM = m
FORM                          PREP = selected
PFRM = m                      PBEG
```

```
PREP = selected
PBEG
.5 L (Fever in {’high’})
.4 L (Fever in {’high’,’low’})
PEND
POTENTIAL
BPCN = Colic
FORM
PFRM = m
PREP = selected
PBEG
.9 L (Fever in {’none’})
PEND
POTENTIAL
BPCN = Infar
FORM
PFRM = m
PREP = selected
PBEG
.6 L (Fever in {’none’})
.3 L (Fever in {’none’,’low’})
PEND

% rules on Abd_pain

BELIEF
BSHN = Abdp_D
BFLN = Abd_pain_Given_Diag
BDVR = Abd_pain
BCVR = Diag
POTENTIAL
BPCN = Appen
FORM
PFRM = m
PREP = selected
PBEG
.6 L (Abd_pain in {’strong’})
.35 L (Abd_pain in {’mild’,’strong’})
PEND
```

```
0.5 L (App in {’yes’}) ...
impl (Fever in {’high’})
0.4 L (App in {’yes’}) ...
impl (Fever in {’high’,’low’})
PEND

BELIEF
BSHN = RCF
BFLN = RColicFever
BDVR = Fever Colic
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.9 L (Colic in {’yes’}) ...
impl (Fever in {’none’})
PEND

BELIEF
BSHN = RIF
BFLN = RInfarFever
BDVR = Fever Infar
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.6 L (Infar in {’yes’}) ...
impl ( Fever in {’none’})
.3 L (Infar in {’yes’}) ...
impl ( Fever in {’none’,’low’})
PEND

% rules on Abd_pain

BELIEF
BSHN = RAA
BFLN = RAppendAbdp
BDVR = Abd_pain App
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.6 L (App in {’yes’}) ...
impl (Abd_pain in {’strong’})
0.35 L (App in {’yes’}) impl ...
(Abd_pain in {’mild’,’strong’})
PEND
```

```
POTENTIAL
BPCN = Colic
FORM
PFRM = m
PREP = selected
PBEG
.9 L (Abd_pain in {’strong’})
PEND
POTENTIAL
BPCN = Infar
FORM
PFRM = m
PREP = selected
PBEG
.8 L (Abd_pain in {’none’})
.15 L (Abd_pain in {’none’,’mild’})
PEND

% rules on Abd_pain

BELIEF
BSHN = Thp_D
BFLN = Thor_pain_Given_Diag
BDVR = Thor_pain
BCVR = Diag
POTENTIAL
BPCN = Appen
FORM
PFRM = m
PREP = selected
PBEG
.95 L (Thor_pain in {’none’})
PEND
POTENTIAL
BPCN = Colic
FORM
PFRM = m
PREP = selected
PBEG
```

```
BELIEF
BSHN = RCA
BFLN = RColicAbdp
BDVR = Abd_pain Colic
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.9 L (Colic in {’yes’}) impl ...
(Abd_pain in {’strong’})
PEND

BELIEF
BSHN = RIA
BFLN = RInfarAbdp
BDVR = Abd_pain Infar
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.8 L (Infar in {’yes’}) ...
impl (Abd_pain in {’none’})
.15 L (Infar in {’yes’}) ...
impl (Abd_pain in {’none’,’mild’})
PEND

% rules on Abd_pain

BELIEF
BSHN = RAT
BFLN = RAppendThorp
BDVR = Thor_pain App
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.95 L (App in {’yes’}) ...
impl (Thor_pain in {’none’})
PEND

BELIEF
BSHN = RCT
BFLN = RColicThorp
BDVR = Thor_pain Colic
POTENTIAL
FORM
```

```
.9 L (Thor_pain in {’none’})
PEND
POTENTIAL
BPCN = Infar
FORM
PFRM = m
PREP = selected
PBEG
.75 L (Thor_pain in {’strong’})
.20 L (Thor_pain in {’strong’,’mild’})
PEND
```

```
PFRM = m
PREP = selected
PBEG
.9 L (Colic in {’yes’}) ...
impl (Thor_pain in {’none’})
PEND

BELIEF
BSHN = RIT
BFLN = RInfarThorp
BDVR = Thor_pain Infar
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.75 L (Infar in {’yes’}) ...
impl (Thor_pain in {’strong’})
.20 L (Infar in {’yes’}) ...
impl (Thor_pain in {’strong’,’mild’})
PEND
```

```
% a priori beliefs on symptoms (Fever and AbdP noisy)
```

```
BELIEF
BSHN = F0
BFLN = Fever0
BDVR = Fever
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.2 L ( Fever in {’low’})
.4 L ( Fever in {’low’,’high’})
PEND

BELIEF
BSHN = AP0
BFLN = Adb_pain0
BDVR = Abd_pain
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.7 L (Abd_pain in {’mild’})
PEND

BELIEF
```

```
BELIEF
BSHN = F0
BFLN = Fever0
BDVR = Fever
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.2 L ( Fever in {’low’})
.4 L ( Fever in {’low’,’high’})
PEND

BELIEF
BSHN = AP0
BFLN = Adb_pain0
BDVR = Abd_pain
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.7 L (Abd_pain in {’mild’})
PEND

BELIEF
```

```
BSHN = TP0
BFLN = Thor_pain0
BDVR = Thor_pain
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
1 L (Thor_pain in {'none'})
PEND

TBMLAB END
```

```
BSHN = TP0
BFLN = Thor_pain0
BDVR = Thor_pain
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
1 L (Thor_pain in {'none'})
PEND

% rule to assert : one diagnosis only

BELIEF
BSHN = DiagUni
BFLN = Diagnosis_Unique
BDVR = App Colic Infar
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
1 L ...
( (App in {'yes'}) and (Colic in {'no'})
and (Infar in {'no'}) ) or ...
( (App in {'no'}) and (Colic in {'yes'})
and (Infar in {'no'}) ) or ...
( (App in {'no'}) and (Colic in {'no'})
and (Infar in {'yes'}) )
PEND

TBMLAB END
```

Let us proceed with the on-line mode for the conditional case.

4.3.3 Phase 1: Variables

There are thus 4 variables:

1. fever : elements = none, low, high
2. abdominal pain : elements = none, mild, strong
3. thoracic pain : elements = none, mild, strong
4. diagnosis : elements = appendicitis, colic, infarction

Even thought there are 4 variables, there are only 3 frames (and 3 structures, one for each frame):

1. a frame with elements : none, low, high. The frame called nlh.
2. a frame with elements : none, mild, strong. The frame called nms.

3. a frame with elements : appen, colic, infar. The frame called diag.

There are 4 attributes (non-indexed) with one variable for each attribute:

1. fever
2. abdominal pain
3. thoracic pain
4. diagnosis

Enter TBMLAB. Type tbmlab in MATLAB Command Window, then RETURN, click on the welcome screen, and wait for the Root Window and its green button.

Select the menu Edit/Variables. A full screen panel pops up, with a comment at left, and the lists of available frames, structures, attributes and variables (none by now of course). The creation order is always frame, structure, attribute, variable as each requires the previous ones.

We create the 3 frames. Select the Frame list by clicking anywhere in the white space of the panel Frame. When a panel is empty and you select it, TBMLAB creates a new object which nature corresponds to the objects listed in the panel.

At bottom left you receive a query dialog box. Give a short name to your frame (erase default name and type in nlh). Leave the full name blank. The fullname is the short name by default. For the ‘variable type’, there is a little triangle at right: it indicates that there is a list attached to the answer box, that you get the list by selecting the answer box with the pointer (button down), and select with the pointer the option Classes_Ordered, just as you would do with a menu.

Enter 3 for the cardinality. End by clicking on the OK button.

You receive at bottom left, a new panel where you can enter the elements labels. The defaults C1, C2 and C3 are to be replaced by the 3 possible values: none, low, high. Do it and end by clicking on the OK button. The frame nlh is created and you can see its name has appeared in the Frame list.

The last panel asks if you want that TBMLAB creates a structure, a non indexed attribute and a variable based on the frame under construction. If you select button Yes, a structure, an attribute and a variable are created based on that frame. It is a very fast trick to build a variable when its only interesting information is the list of its elements. Here we don’t want this option, so you select button No.

To create the second frame, there are two solutions:

- you use the menu Edit/New Frame and you get the query panel for the new object,
- you click in the white area in the Frame panel and you get a panel in which you select New.

Note that if you double click twice quickly on an existing Frame object (one in the list) you get its query panel directly and you can edit the object by modifying the answers to the queries. If you are not fast enough, you get the panel asking to select one object, or asking for a New frame object, or proposing to Cancel the operation.

Select the FRAME panel. Then proceed as with the previous frame. Short name is nms, variable type is Classes_Ordered, cardinality is 3, labels are: none, mild, strong. creating structure... : no.

Finally create the Diag frame. Do it as before with short name = diag, Variable type = Classes_Un_ordered, cardinality 3, labels: appen, colic, infar.

We create the 3 structures. Select the Structure list by clicking in the white area in the Structure panel. Being empty, it means creating a structure. Answer the queries. Choose the short names = Snlh. When asked, select in the frame list the frame nlh on which this structure is built. Then select Done button.

Do it again (using now the New Structure menu or the New button that appears if you double click on the Structure list area) for Snms and then for Sdiag.

Select the Attribute list. Select the Attribute list by clicking in the white area in the Attribute panel. In the query panel, give a name to the new attribute: Fever. Select its structure (Snlh). For number of indexes, there are no indexes, so leave the 0. Select OK.

Proceed identically (starting from the menu New Attribute or the New button that appears if you double click on the Attribute list area) with Abd_pain (structure: Snms), Thor_pain (structure: Snms), Diag (structure: SDiag).

Select Variable. Do as before. In its query panel, give it a short name: Fever. The fact that Variable and Attribute share the same short name is perfectly valid. No confusion can occur.

When you quit the query panel, you must provide an attribute and a frame for that variable. In fact look at the color of the title of the object list. The withe ones are those where you are not supposed to go, and the yellow ones are those where you must do a selection. Select its attribute (Fever), its frame (nlh), then the Done button.

The other three variables are built similarly (using New Variable from the menu or the New button).

When finish, select menu Quit. You are back to the main pull_down menus with the green button.

It is a good idea to save your data. Use File/Save. TBMLAB proposes a default name (data), and presents the Application_Data folder. Create a folder for your application, and then save your data under the name you want. The suffix .obj.mat will be added by the system.

If a file had already been saved during this session, TBMLAB presents directly the corresponding folder and suggests to save data with the folder name. You can accept it or change it at will. Beware that Replace means the data previously saved in that file are destroyed.

In the present case, create a new folder and proceed. I have saved my own data under DiagFTA_CBf name. Later on you may like to load my binary data.

Notes. You have surely noticed the many ‘cancel’ buttons. You can always cancel an operation under way by selecting it. TBMLAB drops the action under way and brings you back where that action started.

Another trick. In the button panels, the button with the strongest red color (at upper left) can sometimes be activated with a return (provided you did not enter a list panel). The lower right button can be activated by the Esc key. If TBMLAB does not answer to your Return and Esc keystrokes (platform dependent), select the button with the pointer.

4.3.4 Phase 2: Beliefs: Rules

You must now enter the conditional belief functions representing your knowledge about the relations between diagnoses and symptoms.

Select Edit/Belief. Select button New. Fill the query panel: short name = Fev_D, full name = Fever_Given_Diag, skip next queries up to last one where replace no by yes. Select the (domain) variable Fever. Select the conditioning variable Diag. Accept or go to the potential window.

Select the element on the conditioning domain for which you want to enter the potential. Select bp 1 Diag=Appen m. In fact a vacuous bba has been created for every element if Diag, what is shown by the m label (other labels could be listed if other forms have been created for the considered potential).

You enter the Potential Control Window. Select the Logical Expression button (upper left panel). You enter into the Bba Editor: logical mode Window. Select successively the buttons Fever, high, Go_On, OK. Enter .5 in bottom right query area. Select button OK. The first mass is allocated. Proceed by selecting the buttons Fever, multi-select (low, high), Go_On, OK, .4, You can then either select OK and menu File/Quit, or the Quit button that does the same task. The unallocated .1 mass is given to the universe. Your first conditional potential has been built.

Select Get next potential in button panel Control. Look at the window title: it states Fever_Given_Diag, bp number = 2. So you know you are working now on the second conditional potential. Use the Logical Expression button and proceed as in the previous case allocating .9 to Fever ’none’. Again use the Get next potential and allocate .6 to Fever ’none’ and .3 to Fever ’none’,’low’.

When back to the Potential Control Window, select quit button in Control panel or in the File menu (both perform the same task).

Repeat the same operation for Abd_pain_Given_Diag

- bp(1) Appen: masses .6 Abd_pain ’strong’, .35 Abd_pain ’mild’,’strong’.
- bp(2) Colic: mass .9 Abd_pain ’strong’
- bp(3) Infar: masses .8 Abd_pain ’none’, .15 Abd_pain ’none’,’mild’.

Finally repeat the same operation for Thor_pain_Given_Diag

- bp(1) Appen: mass .95 Thor_pain ’none’.
- bp(2) Colic: mass .9 Thor_pain ’none’.

- bp(3) Infar: masses .75 Thor_pain 'strong', .20 Thor_pain 'strong','mild'.

Quit and go to the root menus, so you can again save the environment. When TBMLAB asks if we save on the same file as before, accept it as all TBMLAB does is to replace the old files by the new ones, thus everything is saved.

You can also consider a better saving procedure. As all variables have already been saved, you can select the Save data button in the Control panel in Potential Control Window. it is all we need, and we avoid having to go back to the menu window. It is a good and safe procedure to use Save data each time a new belief has been built. Think about possible crashes.

4.3.5 Phase 2: Beliefs: Priors

You must now proceed with the construction of the prior beliefs you may have on some variables. These beliefs are not conditional and built as in the example with logical propositions (section 4.2).

You proceed as for conditional beliefs, except you will not encounter the panel asking to select a potential (as there is only one potential in these cases).

The masses to be entered are

- belief Fever0, .2 Fever 'low', .4 Fever 'low','high',
- belief Adb_pain0, .7 Abd_pain 'mild',
- belief Thor_pain0, 1 Thor_pain 'none'.

4.3.6 Phase 3: Computation

Having saved everything, you want to compute your belief about the diagnoses. Use Tool/Valuation Base Systems (Dir/Undir). When asked which belief object must be used, select the six belief objects listed, then the OK red button. When asked the variables on which marginalization must be computed, select Diag, then the OK red button.

Results are displayed as in the previous example. The pignistic probability given to Infar is .812 with a belief of .675, hence the patient seems to suffer from an Infarction.

4.4 Graphical Method Demo

Data can be displayed as an EVidential Network (EVN). It is a very convenient graphical representation of the belief objects and their domains, enhancing the possible conditional independences between variables. It is inspired by the Bayesian Networks invented by J. Pearl, but completely re-adapted by Shafer and Shenoy, and that fits for any form of uncertainty. We use it for belief function propagations. It works with both joint and conditional beliefs.

Start TBMLAB. Do as already explained.

Load the environment. To help you, the data have already been saved in folder DiagFTA, file DiagFTA_CBf. So to be sure to get the right data, use the saved ones. In the Root Window menus, select File/Open. In the panel, open file DiagFTA_CBf..

Loaded data are verified for syntactical validity. If they were erroneous, you would get a message, and the best attitude would be to recreate the binary files. If they are correct, TBMLAB tells it and you just proceeds.

Select Tools/Graphical Method. Among the Root Window menus, select Tools/Graphical Method. You receive the EVN window. The window is of course still empty. Look at its menus, the dimmed ones are not enabled presently. In fact you must start by entering the variable and belief objects. Select the menu Objects/Get Belief. You receive the list of available belief objects, the six we had created in the previous example.

Multi-select the six belief objects. Multi-selection is done with command-click on Mac or control-click on PC for one-by-one selection or with capital-click for selecting all those in between. Then click on Done. EVN manages to find out the involved variables. A messy network pops up more or less well centered.

The links are not well separated, so we must clean up the graph.

Variables are in grey circles or ovals, beliefs in red hexagons.

Cleaning up the graph. Select menu View/Better Graph. TBMLAB tries to produce a graph were nodes are better located, avoiding overlaps. The result is not so good. So ask for Again (left red button). The resulting graph is quite good now. The operation is iterative. You can still perform a new run by selecting the button Again. The Previous button brings you back to the previous graph; it is useful if the previous graph was better than the new one. The button Enough stops the process and brings you back to the menus. Select it.

If you don't get a nice plot, you can perform the node relocation by hand. You click on the object you want to move to select it, reclick on it to drag and drop it. Links follow the node displacements. For further node displacements, drag and drop works directly. While you work on the node displacements, your menus are not accessible (dimmed). When you are through with the node displacements, click on the upper left red button with Menus Off. It turns green, becomes Menus On, and menus becomes accessible.

Look at the network. Some nodes are ovals/circles painted in grey, other are elongated hexagons painted in pink. The oval/circle-grey ones represent variable nodes. If the variable short_name is short, you get a circle. If the variable short_name is long you get an oval. The pink-hexagons are belief nodes. They represent joint beliefs built on the variables connected to the belief node by an hedge. When a belief corresponds to a joint belief, all links are just lines.

When conditional beliefs are involved, arrows appear. Let $m^{V1,V2,V3}[C1,C2]$ be a bba on $V1,V2,V3$ given $C1,C2$. The link between $C1$ ($C2$) and the belief node representing the bba ends with an arrow, whereas the link from the belief node to the domain variables are just plain lines. So the link between Diag and FevD ens with an arrow, whereas the one between FecD and Fever is just a line.

Recentering the graph . After all these operations you may want to recenter the graph. Use the menu View/EVN at Center, or the menu View/Re Center. The first puts the graph at the center of the window. The second put it where you want.

With the second menu, you get a cross hair. Move it with the mouse, and when you click, what is under the cross hair will be located in the center of the window.

This reorganization of the network can be done for small networks. With larger ones, but not too large (and as far as TBMLAB is a demonstrator, they are never too large, 25 variables being close to the maximum TBMLAB can absorb, you can also enter the belief objects one by one. In fact you enter first a group of them as before, and then add the others one by one or in small groups using Objects/Get Belief each time. It is sometime easier to organize a clean network that way.

TBMLAB protects you against entering twice the same belief object.

Zooming. If you want, you can zoom in and zoom out you network. Select View/Zoom in and/or Zoom out just to see what happens. View/Initial Scale menu redraws the network with the initial scale, and View/Full Network tries to draw large networks on a scale so that the EVN fits within the window limits.

If you choose the menu Objects/New Propositions or New Variables or New Belief, you are send back to the belief/variable creation programs. You just create them. The interest is that the network you have drawn up to now is preserved. Still this is not a good strategy, try to plan ahead and have all your objects ready before moving to the EVN.

EVN propagation. Computation based on the VBS algorithm is performed when you select the menu EVN Control/Propagate.

BetP Histograms. You can then ask to see the values of BetP for every variable present in the network. Select the menu EVN Control/Display BetP. BetP values are presented for each variable as a bar chart. You can switch between BetP and EVN plot by just selecting the appropriate menu in EVN Control, the EVN Control/Display Network or the EVN Control/Display BetP menus.

The BetP histograms are displayed atop of their corresponding variable nodes. Suppose you decide to displace some of the variable nodes, BetP histogram will stay where it was before the node displacement. If you want that the BetP histograms go atop of their corresponding variables, use menu EVN Control/Relocate BetP. The location of the histograms is modified and BetP histograms are now atop of their corresponding variable nodes.

Once BetP plot are available, they can also be moved by hand. Select View/Move Histo. Drag and drop the histo you want to move.

Just remember, to get menus back when light at upper left corner is red, select the red button. It becomes green and menus becomes available.

Nodes Menus. Menus are attached to the nodes. You can access them by selecting a node while holding control key for Mac and PC or using right click on PC.

For the variable nodes, the menus are:

- Show Bel,BetP,Pl: you go to the Display of Belief window, where the only concerned variable is the one of the corresponding node.
- Show list bbm: you get the window with the list of the largest masses.
- Show Belief Editor: you go to the belief potential window.

In each of these functions, menu File/Quit brings you back to the EVN window.

For the belief nodes, the menus are:

- Edit Selected Masses: no C/I: you go to the mass editor, so you can change the masses and the foal sets used by the belief represented by the selected belief node. It may not be used for Continuous or Integers variables (which are not allowed in the EVN program in any case). If the upper left button is red, you must click on it (right-click on PC).
- Show Bel,BetP,Pl: you go to the Display of Belief window, where the concerned variables are those included in the belief object represented by the selected belief node.
- Show list bbm: you get the window with the list of the largest masses, on the domain made of the variables included in the belief object represented by the selected belief node.
- Show Belief Editor: you go to the belief potential window.

In each of these functions, menu File/Quit brings you back to the EVN window.

File menus are:

- Load Network to load previously saved networks. In that case do not use Objects/Get Belief
- Save Network on file.
- Save Objects, what is a good strategy as you probably used the propagate option, and this is a way to save the computed beliefs.
- Print EVN: a way to save the plot for entering it into another file (Latex,...)
- Clear EVN, in which case all you did in the EVN Window is deleted (and lost of course if you did not first save the objects).
- Quit EVN : quit the session and go back to the Root Window with the green button.

Quitting EVN session. As we are through with the example, select this option.

4.5 FTA Diagnosis problem: continue

You can run the batch mode file DiagFTA_CBf or DiagFTA_JBf. Both produce the same results. Just run the DiagFTA_JBf and when the objects have been created, ask for the VBS. You will see that the results are identical to those computed with the conditional beliefs.

There is still a third file that concerns more the General Bayesian Theorem. Run the Batch Mode file DiagFTA_GBT. The objects are those of DiagFTA_CBf, except we did not create the belief object TP0. Several batch mode computations are performed that we study now.

```
COMPUTATION

GBT                        Calling the GBT program
BSHN = Fever_Impact        the short name for the generated belief
CNDB = Fev_D               name of belief with the conditioning potentials
PRBX                       announces next belief object is
BSHN F0                    the belief about the noisy observation
COMP                       ask the GBT computation to be performed

GBT
BSHN = AbdP_Impact
CNDB = Abdp_D
PRBX
BSHN AP0
COMP

GBT
BSHN = ThP_Impact
CNDB = Thp_D
PRBX                       announces next belief object is
SET B 001                  a conditioning on Thp_D is none
% this means NO thoracic pain
COMP

CCR                        calling for the conjunctive combination function
BSHN = Diag_GBT            the short name for the generated belief
BFLN = Diag_Fever_Adbp_Thp
                           its full name
BELF = Fever_Impact, AbdP_Impact, ThP_Impact
                           Those belief to be combined
DISC = 0,0,0               no discounting asked for
```

When computation is over, select menu Edit/Belief, select edit bba button, select Diag_Fever_Adbp_Thp, ask to go to the potential editor and ask for Bel,BetP,Pl,bba. Results are identical to those obtained previously.

4.6 The Best Supported Set Demo

Following an idea of Alain Appriou, we determine the subset of the universe which 'most probably' contains the actual world. For every subset $A \subseteq \Omega$, we compute $Pl(A)/|A|^r$ with $r \in [0, 1]$, and Pl is the normalized plausibility function. For given r, we determine the A that maximizes the criterion.

Value $r = 1$ favors the selection of a singleton, whereas $r = 0$ favors the choice of the universe. Large r favors small sets and vice versa.

When $|\Omega| \leq 12$, the procedure is exhaustive. For $|\Omega| > 12$, the procedure is done according to a forward stepwise procedure.

This task is achieved within the view option in the potential editor.

A set of data have been prepared as a demonstrator. Open the application folder Best_Supported_Set and open the file Var_20_Elem.batch.txt. You receive a variable on a frame with 20 elements, and a belief function built with 512 random masses. Use menu Edit/Belief and select Edit bba, then select B20 in the Belief list, accept the editing panels, and go to the potential editor. In the upper right panel View a potential, select the button Best Supported Set.

The window shows at upper left, the list of the elements of Ω. Each column represents a subset of Ω (the subset made of the elements indicated by a x). Each focal set has a number $1, 2, ...$ and its normalized plausibility is presented. The subsets presented are those with maximal plausibility among those of equal cardinality. So in column 3, the best supported set with 3 elements is the set made of the elements A, G and S. In fact the numerical index of the column is also the cardinality of the selected set.

At bottom, we list for the selected r values, the cardinality (the number) of the best supported set.

At right, we plot the criterion as a function of the cardinality and for those selected r values (indicated at the right end of each curve).

Values of r are selected such that they cover the domain of interest. Values of r smaller then the one used for the lower curve support the same set as the one supported by the r value of the lower curve. Values of r larger then the one used for the upper curve support the same set as the one supported by the r value of the upper curve.

The conclusion as that the best supported set has 3 elements A, G and S, and that the actual world is essentially one of them. The other worlds can be neglected.

4.7 The Chest Clinic Problem

4.7.1 The problem

Shortness of breath (dyspnoea) (D) may be due to either tuberculosis (T), lung cancer (L) or bronchitis (B). A recent visit to Asia (A) increases the risk of tuberculosis, while smoking (S) increases the risk for both lung cancer and bronchitis. The result of a single chest X-Ray (X) does not discriminate between lung cancer and tuberculosis, nor does the presence or absence of dyspnoea.

My patient smokes, I don't believe he went recently to Asia, he complains of dyspnoea but he complains of everything.

4.7.2 Batch-Mode

Data are in file Chest_Clinic.batch.txt, in application folder Chest_Clinic.

```
TBMLAB BEGIN          % instructions start here

FRAME
FSHN = FT
FFLN = Logical
FTYP = Logical
FCRD = 2

STRUCTURE
SSHN = SFT
SFLN = Logical
SMRF = FT

ATTRIBUTE
ASHN = A
AFLN = gone_to_asia
ASTR = SFT

VARIABLE
VSHN = A
VFLN = gone_to_asia
VATT = A
VFRM = FT

ATTRIBUTE
ASHN = S
AFLN = smoking
ASTR = SFT

VARIABLE
VSHN = S
VFLN = smoking
VATT = S
VFRM = FT

ATTRIBUTE
ASHN = D
AFLN = dyspnoea
ASTR = SFT

VARIABLE
VSHN = D
VFLN = dyspnoea
VATT = D
VFRM = FT

ATTRIBUTE
```

```
ASHN = F
AFLN = fever
ASTR = SFT

VARIABLE
VSHN = F
VFLN = fever
VATT = F
VFRM = FT

ATTRIBUTE
ASHN = X
AFLN = X_ray_normal
ASTR = SFT

VARIABLE
VSHN = X
VFLN = X_ray_normal
VATT = X
VFRM = FT

ATTRIBUTE
ASHN = T
AFLN = tuberculosis
ASTR = SFT

VARIABLE
VSHN = T
VFLN = tuberculosis
VATT = T
VFRM = FT

ATTRIBUTE
ASHN = B
AFLN = bronchitis
ASTR = SFT

VARIABLE
VSHN = B
VFLN = bronchitis
VATT = B
VFRM = FT

ATTRIBUTE
ASHN = L
AFLN = lung_cancer
ASTR = SFT

VARIABLE
VSHN = L
```

```
VFLN = lung_cancer
VATT = L
VFRM = FT

BELIEF
BSHN = A0
BFLN = asia_prior
BDVR = A
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.8 L ( not A )
PEND

BELIEF
BSHN = S0
BFLN = smoking_prior
BDVR = S
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
1 L S
PEND

BELIEF
BSHN = D0
BFLN = dyspnoea_prior
BDVR = D
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.6 L D
PEND

BELIEF
BSHN = AT
BFLN = A-¿T
BDVR = A T
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.5 L A impl T
```

```
PEND

BELIEF
BSHN = SL
BFLN = S-¿L
BDVR = S L
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.5 L S impl L
PEND

BELIEF
BSHN = SB
BFLN = S-¿B
BDVR = S B
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.6666666 L S impl B
PEND

BELIEF
BSHN = LTX
BFLN = L+T-¿X
BDVR = L T X
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.9 L ( L or T ) impl X
PEND

BELIEF
BSHN = LTBD
BFLN = L+T+B-¿D
BDVR = L T B D
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
.8 L ( L or T or B ) impl D
PEND
```

COMPUTATION

REPORT % displayed on MATLAB Command Window
TBMLAB END

4.7.3 Phase 1: Variables

The variables are:

- A = Gone_to_Asia (T/F),
- S = Smoking (T/F),
- D = Dyspnoea (T/F),
- F = Fever (T/F),
- X = X_Ray normal (T/F),
- T = Tuberculosis (T/F),
- B = Bronchitis (T/F),
- L = Lung cancer (T/F).

The 8 variables are 8 logical propositions denoted A, S, D, F, X, T, B, L.

Select menu Edit/Short Cuts/Logical propositions. Ask for 8 propositions, and give them the short names A, S, D, F, X, T, B, L. For the full names: either you keep the default ones or you type their full names (avoid spaces, it is NOT a good practice for future readability. Use _ instead.). We suggest Asia, Smoking, Dyspnoea, Fever, X_Ray, Tuberculosis, Bronchitis, Lung_cancer.

Save your new environment using menu File/Save. Give a name to your file (like My_Chest_Clin). All data of this demo will be stored there. I stored the example in Chest_Clin folder.

4.7.4 Phase 2: Beliefs

1: Asia prior: $m(\neg A) = .8, m(\text{universe}) = .2$. It means that you have serious reasons to believe the patient did not go to Asia. You go to the menu Edit, select Belief. In the BELIEF button menu, choose new. In dialog box, give a short name = A0, a full name = Asia_prior. Click OK. Select the variable Asia. As a matter of fact is happens it was the selected one, as by default the first object in the list is selected, so not selecting is also OK here, but not later on. Click Done. TBMLAB asks if you want to go to belief potential window, click yes.

In the upper left panel, choose Sel.elem.: No Cont/Int. You get a list of all possible elements of the universe created with the variables used in THIS belief. Build one focal set by keeping F selected (click on T). At top, select the mass (1. here), fill the box with .8. The remainder must be allocated to the universe, so just quit (menu File/Quit), TBMLAB will perform this allocation. Quit the potential editor window (menu File), and get back to the belief editor window. Repeat the same procedure for the next beliefs.

2: Smoking prior: $m(S) = 1$. This just mean I am sure that the patient is a smoker as I saw him smoking.

3: Dyspnoea prior: $m(D) = .6, m(\text{universe}) = .4$. It means patient complains of dyspnoea, but he is not very reliable, he complains for everything. You proceed identically with dyspnoea, except you enter names D0 and Dyspnoea_prior, select Dyspnoea, and enter .6 on T.

4: Rule Asia implies Tuberculosis: $m(A \rightarrow T) = .5, m(\text{universe}) = .5$. Create the belief object: names are RAT and R(A→T). For the variables, select Asia and Tuberculosis.

Note: Multi_selection achieved on Mac by clicking while holding Command key down (or UpperCase click for all in between data selection), on PC use Control Click. When multi_selection is over, select buttons Done, then Yes. You are back to the Potential Window of THIS belief, choose Logical expressions in upper left panel.

You get: Variable List that lists those variables involved in THIS potential, Logical Operators that lists the allowed operators. You build a focal set by creating a logical expression in the bottom box. You do not type it, you build it with the mouse. So click successively on A, impl, T, OK. A new box appears where you must enter the mass. Type .5 in it. Select OK. Then to say it is over, just select menu File/Quit. Unallocated mass is given to universe. A safe attitude is to select the button save belief, at least the beliefs entered up to now are saved, the variables had been saved previously. Then Select menu File/Quit and you are back to the Belief Editor.

5: Rule Smoking increases Lung_cancer risk. This is translated by $m(S \rightarrow L) = .5, m(\text{universe}) = .5$. The origin of these masses comes form the application of the TBM theory to risk factors and odd ratios encountered in epidemiology (odd ratio = 2). Proceed as at step 4.

6: Rule Smoking increases Bronchitis risk. This is translated by $m(S \rightarrow B) = 2/3, m(\text{universe}) = 1/3$ (odd ration 3). Proceed as at step 4.

7: Rule L or T implies X: $m((L \vee T) \rightarrow X) = .9.m(\text{universe}) = .1$. Proceed as at step 4. Use the logical expression (L or T) impl X .9. Parentheses avoid confusion and is a good practice, even though TBMLAB would accept L or T impl X correctly. But this is not always true, so use parentheses. The only thing you may NOT do is using impl and/or equiv more than once. Iterated implications and equivalence are not understood by TBMLAB in its present version.

8: Rule L or T or B implies D: $m((L \vee T \vee B) \rightarrow D) = .8.m(\text{universe}) = .2$. Proceed as at step 7.

This is all. Your data base is built. Save it either with the save button in the Belief Editor window, or much better, by going back to the main menu, and using menu File/Save in which case everything is (re)saved. The Save option in in the Belief Editor window saves only the belief objects, not the variable objects, etc..., whereas File/Save saves everything that exists. Still a better practice is building a belief one by one, and when back to the Belief Editor window, apply Save for each belief. So what has been entered is protected against a crash.

Mistyping logical expressions. If you enter logical expressions incorrectly, it is easy to correct them before accepting them and their masses. In fact, the buttons only create a list of characters, those you see in the box. TBMLAB just reads the content of the box, and parses it. So you can modify the content of the box by typing directly into it, provide you respect the syntax you see. Erasing or adding characters is permitted, but be sure to leave blanks on both side of the operators. This method is correct except it is dangerous if you make an error. The parser is not very smart and a crash might occur. A much safer procedure is to end the construction with the Clear Expr. button. It deletes what you have just created and is not correct, and you restart the construction of the logical expression.

4.7.5 Phase 3: Computation

Select the menu Tool/Valuation Base System. TBMLAB asks you to select the belief objects to be combined. Select the 8 belief objects you have just created. Select OK. Then TBMLAB asks you on which frame the belief should be marginalized. Select the variables on which you want to see the projection of the combined beliefs. A good selection here is L, T, B. TBMLAB does the computation and goes directly to the result display window, where you get many information on the computed belief.

4.7.6 Using Graphical Method

If you like the graphical method for displaying data, enter Menu Tools/Graphical Method. In the new window, select the menu Objects/Belief, and select the 8 first belief objects. You get a messy EVN (Evidential Network).Select View/Better Graph. Use the again button 4 times and you get a nice plot.

Select menu EVN Control/Propagate. Once the beliefs have been propagated, ask for EVN Control/Display BetP. You get BetP for each variable in bar chart representation. You can move the bar charts (using menu View/Move histo and then drag and drop the histos), and see the EVN back. When through, select menu File/Quit EVN and you are back to the root window.

Comment: in the EVN plot, if you select a variable node with right click (PC) or CTRL_click, you get three menus which permits you to display the Bel, BetP, Pl plot, to display the list of bba's, or the send you in the belief editor (in which case you could examine the masses).

The same procedure applied to the belief nodes produces, atop of the last three options, an Edit Selected Mass.

Note that the Graphical Method does not work with Continuous or Integers variables.

4.8 The flight crew problem

4.8.1 The problem

In a certain flight crew, the position of pilot, copilot and flight engineer are held by three persons, Allen, Brown and Carr, though not necessary in that order. (This fact is sure: mass = 1.)

The copilot who is an only child, earns the least. (This rule becomes Copilot → (Only Child and Earn Low) (use -> in TBMLAB). It has a mass .6, reflecting that it is a quite dubious assertion)

Carr, who married Brown's sister, earns more than the pilot does. Brown is not only child, (mass .9), Carr is not the pilot and his earning is not low (mass .8)

What position does each of the three persons hold?

4.8.2 Batch-Mode

Data are in file Flight_Crew.batch.txt.

```
TBMLAB BEGIN

FRAME
FSHN = Occ
FFLN = Occupation
FTYP = Classes_Un_ordered
FCRD = 3
ELEM = Pilot,Copilot,Engineer

FRAME
FSHN = Ear
FFLN = Earning
FTYP = Classes_Un_ordered
FCRD = 3
ELEM = Low,Medium,High

FRAME
FSHN = OnCh
FFLN = Only_Child
FTYP = Classes_Un_ordered
FCRD = 2
ELEM = No,Yes

FRAME
FSHN = Person
FFLN = Individual
FTYP = Classes_Un_ordered
FCRD = 3
ELEM = Allen,Brown,Carr

STRUCTURE
SSHN = Occ
```

```
SMRF = Occ

STRUCTURE
SSHN = Ear
SMRF = Ear

STRUCTURE
SSHN = OnCh
SMRF = OnCh

STRUCTURE
SSHN = Person
SMRF = Person

ATTRIBUTE
ASHN = Occ
ASTR = Occ
INDEX                 % creating an indexed variable
AIXN = Person         % its name
AIXF = Person         % its frame

ATTRIBUTE
ASHN = Ear
ASTR = Ear
INDEX
AIXN = Person
AIXF = Person

ATTRIBUTE
ASHN = OnCh
ASTR = OnCh
INDEX
AIXN = Person
AIXF = Person

VARIABLE
VSHN = Occ
VFLN = Occupation
VATT = Occ
VFRM = Occ
MAKE_ALL_VARIABLES    % one for each possible index value

VARIABLE
VSHN = Ear
VFLN = Earning
VATT = Ear
VFRM = Ear
MAKE_ALL_VARIABLES

VARIABLE
```

```
VSHN = OnCh
VFLN = Only_Child
VATT = OnCh
VFRM = OnCh
MAKE_ALL_VARIABLES

BELIEF
BSHN = Const1Work
BFLN = Constraint_One_Work
BDVR = Occ(Allen) Occ(Brown) Occ(Carr)
BDCR = 3 3 3
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.8 L ...
( (Occ(Allen) in {’Pilot’}) and (Occ(Brown) in {’Copilot’}) ...
and (Occ(Carr) in {’Engineer’} )) or ...
( (Occ(Allen) in {’Pilot’}) and (Occ(Brown) in {’Engineer’}) ...
and (Occ(Carr) in {’Copilot’} )) or ...
( (Occ(Allen) in {’Copilot’}) and (Occ(Brown) in {’Pilot’}) ...
and (Occ(Carr) in {’Engineer’} )) or ...
( (Occ(Allen) in {’Copilot’}) and (Occ(Brown) in {’Engineer’}) ...
and (Occ(Carr) in {’Pilot’} )) or ...
( (Occ(Allen) in {’Engineer’}) and (Occ(Brown) in {’Copilot’}) ...
and (Occ(Carr) in {’Pilot’} )) or ...
( (Occ(Allen) in {’Engineer’}) and (Occ(Brown) in {’Pilot’}) ...
and (Occ(Carr) in {’Copilot’} ))
PEND

BELIEF
BSHN = ConstAllen
BFLN = Constraint_Allen
BDVR = Occ(Allen) Ear(Allen) OnCh(Allen)
BDCR = 3 3 2
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.8 L (Occ(Allen) in {’Copilot’}) impl ( (Ear(Allen) in {’Low’})...
and (OnCh(Allen) in {’Yes’} ))
PEND

BELIEF
BSHN = ConstBrown
BFLN = Constraint_Brown
BDVR = Occ(Brown) Ear(Brown) OnCh(Brown)
BDCR = 3 3 2
```

```
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.9 L (Occ(Brown) in {’Copilot’}) impl (( Ear(Brown) in {’Low’}) ...
and (OnCh(Brown) in {’Yes’} ))
PEND

BELIEF
BSHN = ConstCarr
BFLN = Constraint_Carr
BDVR = Occ(Carr) Ear(Carr) OnCh(Carr)
BDCR = 3 3 2
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.9 L (Occ(Carr) in {’Copilot’}) impl (( Ear(Carr) in {’Low’}) ...
and (OnCh(Carr) in {’Yes’} ))
PEND

BELIEF
BSHN = CSr
BDVR = Occ(Carr) Ear(Carr)
BDCR = 3 3
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
0.7 L (Ear(Carr) in {’Medium’,’High’}) and (Occ(Carr) ...
in {’Copilot’,’Engineer’})
PEND

BELIEF
BSHN = OnlyChBrwon
BDVR = OnCh(Brown)
BDCR = 2
POTENTIAL
FORM
PFRM = m
PREP = selected
PBEG
1.0 L (OnCh(Brown) in {’No’})
PEND

COMPUTATION
```

```
REPORT

VBS
BSHN = WhoWhatAll
BFLN = Who_does_What
BELF = Const1Work,ConstAllen,ConstBrown,ConstCarr,CSr,OnlyChBrwon
PVAR = Occ(Allen),Occ(Brown),Occ(Carr)

VBS
BSHN = WhoWhatAllen
BFLN = Who_does_What
BELF = Const1Work,ConstAllen,ConstBrown,ConstCarr,CSr,OnlyChBrwon
PVAR = Occ(Allen)

VBS
BSHN = WhoWhatBrown
BFLN = Who_does_What
BELF = Const1Work,ConstAllen,ConstBrown,ConstCarr,CSr,OnlyChBrwon
PVAR = Occ(Brown)

VBS
BSHN = WhoWhatCarr
BFLN = Who_does_What
BELF = Const1Work,ConstAllen,ConstBrown,ConstCarr,CSr,OnlyChBrwon
PVAR = Occ(Carr)

PRINT
FILE = Fl_Crew_Comp_Results
BELF = all

TBMLAB END
```

4.8.3 Phase 1: Variables

We have 3 attributes, Oc, Ea, C1, (for Occupation, Earning, Only Child), each with one index = Pr (person) which domain is A, B, C (for Allen, Brown, Carr).

The FRAMES. I need 4 frames, one for each of Oc, Ea, C1, Pr. I name them FOc, FEa, FC1, FPr. Select menu Edit/Variables.

Select the FRAME list. Enter successively: Name = FOc, type = Classes_Unordered, cardinality = 3. OK. In element names, enter Pil, Cop, Eng. No for make variable.

Select the Edit/New Frame, Enter: Name = FEa, type = Classes_Unordered, cardinality = 3. OK. In element names, enter Lw, Md, Hi. No for make variable.

Select the Edit/New Frame, Enter: Name = FC1, type = Classes_Unordered, cardinality = 2. OK. In element names, enter, N, Y. No for make variable, OK.

Select the Edit/New Frame, Enter: Name = FPr, type = Classes_Unordered, cardinality = 3. OK. In element names, enter A, B, C. No for make variable.

The 4 frames have been built.

The STRUCTURE. Select the STRUCTURE list, Enter : Name = SFOc, select the FOc frame, OK.

Select the Edit/New Structure. Enter : Name = SFEa, select the FEa frame, OK.

Select the Edit/New Structure. Enter : Name = SFC1, select the FC1 frame, OK.

Select the Edit/New Structure. Enter : Name = SFPr, select the FPr frame, OK.

The 4 structures have been built.

The ATTRIBUTES. Build 3 attributes called AOc, AEa, AC1, each with one index = Pr, and their corresponding structure.

Select the ATTRIBUTE list. Enter : Name = AOc. OK. Select its Structure = SFOc. Enter number of indexes for this attribute = 1. OK. Index name = Pr. Select in FRAME the frame FPr on which this index is defined. OK.

Do the same for the other two attributes using Edit/New Attribute. The 3 attributes have been built.

The VARIABLES. We have 3 indexed attributes. For each of them, we build all the variables (one for each attribute × index value) in ONE instruction. Here you get 3 variables in one instruction. It works for any number of indexes.

Select the VARIABLE list. Enter : Oc, keep the 'yes' in creating variables for every index. Select the frame on which the variable is defined (FOc). Select the Attribute used to build this variable (AOc). OK.

Quit the Variable Editor using the menu File/Quit.

4.8.4 Phase 2: Beliefs

Select menu Edit/Belief. Button new.

Enter : Names = ClW. Multi_select variables Oc(A), Oc(B), Oc(C). Accept to go to the Potential Editor.

Select Sel. elem.: No Cont/Int. You get the list of all possibilities. As initially you get a vacuous bba, all elements have been selected (as indicated by the x). Click on every x except those you want to keep. You want to keep those elements that fit with the constraint that one person can hold only one job. Keep a x on those line with (Pil, Cop, Eng), (Pil, Eng, Cop), (Cop, Pil, Eng), (Cop, Eng, Pil), (Eng, Pil, Cop), (Eng, Cop, Pil). Mass = .8. RETURN. Select menu File/Quit.

We must now build 3 times the same belief. So we use Duplicate → index option that reproduces automatically the same belief for all possible combinations of the index of its variables. It corresponds to generating : for all x, m(Oc(x) in U)=.7, m(Oc(x) in V)=.3 as many time as there are x's. All we need is just to build one of them, and use the duplicate → index option. So we build a B0 belief for Allen. It will be among those built automatically so the names are consistent. So you can delete B0 once duplication is done or just keep it, it is not important.

Select New: Names = B0. Multi_select variables Oc(A), Ea(A), C1(A). Go to Potential Editor window. Select Logical expression. Logical expressions are fed in by clicking on the operators, the variables and when these are not logical, by selecting the value that it can take (you get an extra list just for that). You enter with mouse selection :

Oc(A) Cop impl (Ea(A) Lw and C1(A) Y) OK. In the new box,enter mass .6. OK. Quit and Save.

For Brown and Carr, use option Duplicate → index. You get one for Allen also as said before.

Select menu Edit/Belief. Button new. Enter : Names = CSt. Multi_select variables Oc(C) and Ea(C). Accept to go to the Potential Editor. Select Logical expressions. Ea(C), Md, Hi, OK (at UL) , and Oc(C) Cop, Eng, OK (at UL) ,OK,.7, OK,Quit and Save. Back to belief.

Select menu Edit/Belief. Button new. Enter : Names = BrOC, Select C1(B), Accept to go to the Potential Editor. Select sel.elem.: 1 mass. Select N, mass = .9, mass OK, bba OK. Quit and Save. Back to belief. Save belief. Quit.

The data base is built and saved.

4.8.5 Phase 3: Computation

Done as in the previous examples using VBS.

4.9 BetP from likelihoods, continuous domains, large cardinality.

4.9.1 The problem

Let T be s set of 100 possible targets. Let X be an observable variable. For each $t_i \in T$, the belief about X is given by a probability density function (pdf) like a gaussian distribution which mean and standard deviation depend on the target.

Suppose you collect an observation on the target and compute the likelihood for each target, a vector with 100 likelihoods.

We assume a vacuous a priori on T. We also assume that the pdf represents the user's pignistic probabilities?

What is the most probable target we are facing, given the likelihoods and the pdf.

The interest of this example is to show that we can handle large frame. There are of course programming tricks, but this true for many applications. Size is an issue for which one must find solutions, not just run away.

The batch mode file Test_GBT_pdf.batch.txt contains the needed data. Entering the 100 pdf by hand is very tedious. It is done easily as we can run a .m function within the batch mode run (see RUN).

4.9.2 The Batch Mode File

```
TBMLAB BEGIN
% to change cardinality ,
% just change the FCRD value in the FRAME object below

% un_ordered classes
```

```
FRAME
FSHN = T
FTYP = Classes_Un_ordered
FCRD = 100                change here for larger frames
DO_VAR                    related structure, attribute and variable created

% a continuous variable
FRAME
FSHN = X
FTYP = Continuous
ELEM = -inf,inf           inf or infinity or infinite mean it
DO_VAR

% conditional belief on X (integers, univariate) given T (discrete, univariate)

% case pdf is bbf
BELIEF
BSHN = XT
BDVR = VX
BCVR = VT
RUN                       You run an external .m function
generate_100_pdf('shifted')

% you can use 'random' or 'shifted'
% the labels for T frame elements are defined in the generate_100_pdf function

COMPUTATION

% REPORT

GBTPDF                    run the GBT with pdf
BSHN = GBT                name of object with results
CNDB = XT                 name of the belief object with conditional beliefs
DATA = 24,27              conditioning event: data is between 24 and 27

TBMLAB END
```

You can edit the file and replace 'shifted' by 'random'.

With the shifted option, the means are 15, 20, 25, ...510. The standard deviation is 10.

With the random option, the means are random integers between 0 and 100. The standard deviations are random integers between 1 and 30.

All you have to do is to run tbmlab and to open the Test_GBT_pdf.batch.txt file.

Be patient, building all the potentials is slow (30 sec) whereas the BetP computation itself takes less that 1 sec. In fact the purpose of the example was to show that large cardinalities can be handled provided programming is smartly done. Brute force programming leads nowhere in such large problems, but preliminary reflexions permit drastic reductions of the computation load.

You get the results in a special window that displays 4 parts.

1. At upper right, a list with the parameters of the 100 pdf.
2. At lower right, the list of the 10 most probable targets.
3. At upper left the Bel, BetP, Pl limits for the 10 most probable targets.
4. At lower left, the pdf of the 5 most probable targets. The color codes are listed in the lower right box. The X observation is represented by the gray area.

The help menu allows you to get a few hints about what you see.
To proceed, select menu File/Quit.

Chapter 5

Batch Mode Syntax

Details of the instruction used for entering data in batch mode. Instructions are self contained or need extra data.

5.1 Instruction Lines

The text file with the batch-mode instructions is made of lines.

All **blanks at the beginning or at the end** of a line are skipped. This allows a nice layout of your text file with indents (but don't use tab).

When the **instruction is too long** to fit into one line, end the line with ... (3 dots) and the next line becomes a continuation of the previous one.

The **first line** must be TBMLAB BEGIN. The **last line** must be TBMLAB END. Whatever is not between them is skipped. This is needed to get rid of the header and tail created by some text editors.

Lines can be instruction lines or comment lines.

- **Comment lines** start with the % symbol. They can be put anywhere, as they are skipped by TBMLAB.
- **Empty lines** are also considered as comments and are thus ignored.
- **Self contained instruction lines** are made of one word (using _ sometimes).
- **Data instruction lines** start with an instruction (usually a four letter instruction) followed by a space, an equal sign, a space and data.

Instruction can be in capital or lower-cases, TBMLAB does not differentiate.

In the MATLAB Command Window, each instruction line is printed just before being decoded. So in case of **crash**, go to the MATLAB Command Window and look at the last printed instruction line. It is the instruction that causes the crash. Correct it in your batch mode instruction file.

For MATLAB programmers: each line (except the self contained instruction lines) is evaluated with the eval function after some extra symbols have been added, like [ ], ' ' or {}.

The file Matrix_Maximal (located in User_Data/Batch_Mode_Matrices) lists all the objects needed for creating the batch_mode input file. It can seriously help the user who can build each object using copy paste. Instructions in capital are usually compulsory, those in lower-cases are optional.

The file Matrix_Minimal (located in User_Data/Batch_Mode_Matrices) lists all the objects really needed for creating the batch_mode input file.

The file Test_Batch.batch.txt contains many instructions I use to test the Batch-mode program. They can be read as illustrations of what can be done in Batch-Mode.

5.2 Saved files use .batch field

The batch_mode function opens the folder User_Data/Application_Data. For simplicity sake, the batch mode file is suffixed with .batch.txt.

To open a file, select and Open it.

5.3 The self contained instructions

The self contained instructions are one word instructions, thus without an = sign and data. Some are detailed later within their specific domain.

- TBMLAB BEGIN : this instruction must be in front of your data, whatever is before is skipped. If missing, nothing done.
- TBMLAB END : this instruction must be at the end of your data, whatever is after is skipped.
- FRAME : announces that a new frame is to be created using the instructions that follow it.
- DO_VAR : asks for the automatic construction of a variable (plus structure and attribute) based on the frame where the instruction is located.
- STRUCTURE : announces that a new structure is to be created using the instructions that follow it.
- MAPPING : NA details the relation that exists between two frames that belong to the same structure.
- ATTRIBUTE : announces that a new attribute is to be created using the instructions that follow it.
- INDEX : announces that what follows concerns one index of the attribute under construction.
- VARIABLE : announces that a new variable is to be created using the instructions that follow it.
- MAKE_ALL_VARIABLES : builds one variable for every possible value of the indexes of its attribute.
- FINISH_IT : for variables without indexes, you can just create its FRAME (let its short name be Fr), give to the variable a short name (say VarX) and optionally a full name (say VariableX), assign its frame (Fr here) and use

the instruction FINISH_IT. In that case TBMLAB creates a Structure which short name is SVarX, long name is SVariableX and most refined frame is Fr, and a non indexed Attribute which short name is AVarX, long name is AVariableX and structure is SVarX.

- BELIEF : announces that a new belief is to be created using the instructions that follow it.
- POTENTIAL : announces that you ask for a new potential for the belief object under construction.
- FORM : announces that you ask for a new form for the potential object under construction.
- RANDOM : generating a random bba (at most 512 focal sets).
- RANDOM_N : generating a random normalized bba (at most 512 focal sets).
- PBEG and PEND : instructions in between creates the potential values (like the basic belief masses).
- PRDF : announces next line create pdf which may be one of {binomial, beta, gamma, gaussian, laplace}
- COMPUTATION : announces that computation must be performed on some belief objects.
- VBS : announces that the computation to be performed is to be executed with the Valuation Based System.
- CHANGE_FRAME : announces that you ask for a marginalization and/or vacuous extension of a bba stored as an m, wc or wd function.
- GBT : announces that the computation to be performed is the GBT (from observation to parameter).
- DRC : announces that the computation to be performed is the DRC (from parameter to observation).
- PRBT : announces next data are relative to the parameter space (for GBT, DRC).
- PRBX : announces next data are relative the the observation space (for GBT, DRC).
- COMP : asks that GBT or DRC computation is performed now.
- CCR : announces that the computation to be performed is the conjunctive combination rule.
- DCR : announces that the computation to be performed is the disjunctive combination rule.
- CAUTIOUSCC : announces that the computation to be performed is the cautious conjunctive combination rule.

- GBTPDF : announces that the computation to be performed is the General Bayesian Theorem where the input are probability density functions (pdf), and the apriori in vacuous.
- RUN : announces that the next line is a MATLAB instruction xxx to be executed by eval(xxx) (requires knowledge of MATLAB and TBMLAB languages, but can be very convenient).
- PRINT : announces that some printing is asked for.
- REPORT : asks that a report is displayed on the MATLAB Command Window.
- LOAD : load a binary file previously saved, file name asked on the screen.
- SAVE : save all objects in a file which name in asked on the screen
- SKIP VERIFICATION : normally loaded data are verified, but you an skip this verification phase with the present instruction.

5.4 The instructions with data. Object definition.

These instructions depend on the object under construction. Every object must have one short_name, which will be used to refer to it. Short names must be unique. Use short and smart Short_names.

Each object also has a Full_name, which by default is equal to the Short_name. Use long name so you can remember what the object is about by reading its Full_name.

Full_name MUST be defined after the Short_name. So be careful to define first a Short_name and then a Full_name if you want a Full_name different from the Short_name.

Optional instructions end with (opt).

Spaces are NOT allowed in names. Use _.

When an object refers to another object, you use the other object short_name.

5.4.1 FRAME

FSHN = short_name
FFLN = full_name (opt)
FTYP = frame_type
FCRD = frame_cardinality (may be opt)
ELEM = element_list (opt)
DO_VAR (opt)

FTYP must be one of:

Logical
Binary
Classes_Un_ordered
Classes_Ordered
Integers
Continuous

The frame_cardinality and element_list input depend on the FTYP value.

If FTYP = Logical,	FCRD and ELEM line skipped
If FTYP = Binary,	ELEM = elem1, elem2
If FTYP = Classes_Un_ordered	ELEM = elem1, elem2, . . . ,elemK
If FTYP = Classes_Ordered	ELEM = elem1, elem2, . . . ,elemK
If FTYP = Integers,	ELEM = lower_limit, upper_limit
If FTYP = Continuous	ELEM = lower_limit, upper_limit

Default values are :

If FTYP = Logical,	FCRD = 2, ELEM = 'F','T'
If FTYP = Binary,	FCRD = 2, ELEM = '0','1'
If FTYP = Classes_Un_ordered	ELEM = A,B...Z,a,b,...z if FCRD $\leq$ 52
If FTYP = Classes_Un_ordered	ELEM = H1,H2.... if FCRD > 52
If FTYP = Classes_Ordered	ELEM = C1,C2...
If FTYP = Integers,	ELEM = 1,2,... FCRD
If FTYP = Continuous	ELEM = 0, 100

To enter infinity for continuous variables, you can use any of inf, infinity or infinite, with a minus sign in front is negative. They are synonymous. Don't be surprised, infinity is equated to 1E40 in the objects.

If the self contained DO_VAR instruction is present, TBMLAB builds automatically:

- a structure with the present frame as its unique frame
- an attribute without index and which structure is the one just created
- a variable based on the just created attribute and which frame is the just created frame

If FSHN (frame short_name) is xyz, then the names are Sxyz, Axyz and Vxyz, respectively.

Example. Some frames.

FRAME
FSHN = Int
FFLN = Int_0_119
FTYP = Integers
FCRD = 120
ELEM = 0, 119
DO_VAR

FRAME
FSHN = F_T
FTYP = Logical

FRAME
FSHN = C3
FTYP = Classes_Un_ordered
FCRD = 3
ELEM = red,yellow,black

5.4.2 STRUCTURE

SSHN = short_name
SFLN = full_name (opt)
SMRF= fref the frame reference of the most refined frame.
SSFR = fref1,fref2,...,frefN (opt)
the frame references of the other frames.

Example. The structure concerns the age of a person. Frame Int are integers from 0 to 119. Frame = age2 is ‘Young’, ‘Not Young’. Frame = age10 = 0, 1, 2... 11 (age in decade).

STRUCTURE
SSHN = SAge
SFLN = age_structure
SMRF= Int
SSFR = age2,age10

5.4.3 MAPPING

Not available.

5.4.4 ATTRIBUTE

ASHN = short_name
AFLN = full_name (opt)
ASTR = sref the structure reference
INDEX (opt)
AIXN = index_name the name of the index (only if INDEX present)
AIXF = fref the reference of the frame underlying the index (only if INDEX present)

If attribute has no indexes, skip the INDEX, AIXN and AIXF lines.

For each index, repeat INDEX and use the AIXN, AIXF pair of instructions, in that order.

Index may not be defined with a continuous frame.

Example. The attribute is the income of an individual (frame FIndiv) during year (frame Fyear)

ATTRIBUTE
ASHN = inc
AFLN = Income
ASTR =Inc
INDEX
AIXN = individual
AIXF = FIndiv
INDEX
AIXN = year
AIXF = Fyear

5.4.5 VARIABLE

VSHN = short_name the short name of the variable
VFLN = full_name the full name of the variable (opt)
VATT = aref the reference of its attribute
VFRM= fref the reference of its frame
VINX = value_index1, value_index2, ..., value_indexK
(only if attribute has indexes)
MAKE_ALL_VARIABLES builds a variable for every possible value
of the indexes of its attribute (opt)

If the attribute has indexes, you must provide the values taken by the indexes, what is done by VINX. If the attribute has no index, skip VINX. VINX is a list of numbers indicating, for each index and in the same order as the indexes are defined in attribute VATT, which element of the frame of the index must be used. It points to the position of the element in the element list of the frame. For integer frames, value_index is the numerical value taken by the index. Index may not be defined with a continuous frame.

Example. The variable is the income of John during year 1999.

VARIABLE
VSHN = Inc(J,99)
VFLN = income of John in 1999
VATT = AInc
VFRM= FInc
VINX = 3,9

5.4.6 BELIEF

BELIEF
BSHN = short_name
BFLN = full_name (opt)
BAGN= agent_name (opt) a string with the label of the belief holder
BTME= time (opt) a number indicating time when belief held
BEVC = evidential_corpus (opt) a string expressing it
BDVR = vref11,vref12, . . . ,vref1K
list of variable references,
belief domain is cartesian product of the domains of these variables,
BDCR = card1,card2... (opt) cardinalities of each domain variable
BCVR = vref21,vref22, . . . ,vref2K (opt)
list of variable references,
conditional belief built for each element of cartesian product
of the domain of these variables
BCCR = card1,card2... (opt) cardinalities of each conditioning variable
BDIS = disposable (opt) belief can be deleted
when you run menu Clear unused beliefs.
BCMP= how_belief_was_computed (opt)

The easiest way to enter the focal sets is by using the menu EDIT/Belief. By default this is what user should do. Otherwise he must know how the subsets are represented.

There is a potential for every possible element of the BCVR domain (only one if BCVR is missing, thus we have a marginal belief, not a conditional belief).

POTENTIAL
BPCN = value_vref1,value_vref2, ... ,value_vrefK (opt)
    value of the element in the frame of vref, same order as in BCVR
    Skipped if BCVR absent
FORM
PFRM = mv   for mv ∈ {m, b, bel, pl, q, wc, wd, BetP}
PREP = prep   for prep ∈ {ordered, selected}. See focal set generation
PBEG   announces that what follows are potential values.
one_mass and its focal set
one_mass and its focal set
...
PEND   announces instructions to enter values and focal sets are finished.

BPCN has the form BPCN = Male, Yes, ToDay, 44, 1. Thus the element values, not their positions (spaces are irrelevant, coma is essential, order is essential, same as in BCVR).

The one_mass instruction line can have two forms:

- val L text : where val is the potential value (type as announced by PFRM), L means logical, and text is a logical instruction.
- val S number : where val is as above, S can be B, O, D or X. It tells the system that the next data is a binary, octal, decimal or hexadecimal number, and number is the value of the focal set. Using this option requires a good understanding of the position of the elements in the list of elements (see section 6.9).

If an one_mass instruction line ends with ... (3 dots), it means the next line is a continuation line.

If PFRM = m, the unallocated mass (1 minus those introduced as val) is given to the universe.

We can have several FORM in one POTENTIAL and as many POTENTIAL as elements in BCVR domain (the conditioning domain). The missing POTENTIAL in the conditional case are vacuous belief functions (set by default).

For conditioning potentials, you enter as many POTENTIAL as there are conditioning elements, finishing each by PEND.

Special options.

Random bba. You can generate random bba's, which are useful when learning the TBM. A random bba consist in at most 512 focal sets randomly selected and each receiving a random mass (the sum constraint being satisfied). The simplest form consists in putting RANDOM or RANDOM_N just behind FORM, and a PEND to announces that this belief object is built. The bba is normalized in the second case ($m(\emptyset) = 0$).

POTENTIAL
FORM
RANDOM
PEND

Probability density function When the frame of the variable is Integers or Continuous, You can entered its bba as a probability density function (pdf).

You can choose a pdf among the set {binomial, beta, gamma, gaussian, laplace}. In that case you must enter their nature, parameters and meaning.

- pdf_nature: one of {binomial, beta, gamma, gaussian, laplace}.
- parameters: one of {ab, ms}. ab for the distribution parameters, ms for its mean, and standard deviation. The parameters are those of the Handbook of Mathematical Functions of (Abramowitz & Stegun, 1965).
- meaning: one of {bbf, Betf}. bbf means the pdf is the user's belief function. Betf means the pdf is the pignistic transformation of the user's belief function. In that case, TBMLAB uses the q-least committed (q-LC) isopignistic belief functionas the user's belief function. It means: we build the set of all the belief functions whose pignistic probability function is the given pdf (isopignistic). Among them we select the one that has the largest value for every commonality (q-LC).

POTENTIAL
FORM
PRDF
pdf_nature option1 option2 x_1 x_2
PEND

The instruction line is 'pdf_nature option1 option2 x_1 x_2' where

- pdf_nature is one of {binomial, beta, gamma, gaussian, laplace},
- option1 is one of {bbf, Betf}. bbf means the entered pdf represents the user's beliefs. Betf means the pdf is the pignistic transformation of the user's underlying beliefs, and we know nothing else about these underlying beliefs.
- option2 is one of {ab, ms}. It means that the parameters entered are the ab parameters or the mean, standard deviation parameters.
- x_1 and x_2 are two reals (the values of the 2 parameters announces by option2

5.5 The instructions with data. Ordering computations.

These instructions are entered after the COMPUTATION instruction has been entered.

5.5.1 VBS

To execute the VBS algorithm, all you need are the list of belief objects to be used and the list of variables on which the result must be projected. It generates one new belief object that stores the projection on the variables in the variable list of the result of the propagation of all the belief stored in the list of belief objects.

If some belief objects correspond to conditional bba's, say $m^X[\theta]\ \forall\theta \in \Theta$, the function creates for each belief object a new belief objet $m^{X,\Theta}$on the $X \times \Theta$ domain which is the conjunctive combination of the ballooning extensions of the $m^X[\theta]$:

$$m^{X,\Theta} = \textcircled{\cap}_{\theta\in\Theta} m^X[\theta]^{\Uparrow X\times\Theta}.$$

This construction transforms the conditional bba case into a joint bba case, on which the VBS is applied.

VBS
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
BELF = bshn1,bshn2,...,bshnN
    a list of belief short_names, those to be included in the VBS.
PVAR = vshn1,vshn2,...,vshnK
    a list of variable short_names.
    Marginalization done on their product space.

5.5.2 CHANGE_FRAME

Given a marginal belief potential defined on space X, you can project it on a new space Y, where a space is the cartesian product of several variables. The potential is marginalized on space $X \cap Y$ and then vacuously extended on space Y. Plain marginalization is achieved when $Y \subseteq X$, and plain extension is achieved when $X \subseteq Y$. The operation corresponds to the uparrow in Shenoy's notation. It means we compute $m^{X\uparrow Y}$.

This function requires that the initial potential has at least one of $\{m, wc, wd\}$. You enter the name for the new belief object defined on Y, the name of the initial belief defined on X, and the short names of the variables included in Y.

CHANGE_FRAME
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
BELF = bshn
    the initial belief short_names.
PVAR = vshn1,vshn2,...,vshnK
    a list of variable short_names.

5.5.3 GBT

To execute the GBT algorithm, all you need is a belief object with conditional beliefs and a conditioning event which can be either a subset of the domain or a potential of the domain. An a priori belief on the conditioning domain is allowed.

The function generates a new belief object containing the potential generated by the GBT, with names BSHN, BFLN.

GBT
BSHN = a_short_name

BFLN = a_full_name, = to BSHN is absent.
CNDB = bshn
the belief short_name where the conditional belief is stored.
PRBT (opt) announces that what follows concerns the conditioning domain T
BSHN bel_shn / SET mode focal_set_description
PRBX announces that what follows concerns the domain X
BSHN bel_shn / SET mode focal_set_description
COMP announces that computation is to be performed.

The instructions BSHN bel_shn / SET mode focal_set_description can be one of:

- BSHN bel_shn provides the short_name of the belief object that contains either the a priori belief on T (case PRBT) or the belief about a dubious observation (case PRBX)
- SET mode focal_set_description where mode is one of B, O, D, X and focal_set_description is the binary, octal, decimal, hexadecimal representation of the set on which conditioning (either on T or on X) is performed.

User does not have to enter PRBT data. He must provide the PRBX data.
LIMITATION: the domain of bshn may contain only one variable:
(|belief(.).domain_variable| = 1).

5.5.4 DRC

To execute the DRC algorithm, all you need is a belief object with conditional beliefs and a conditioning event which can be either a subset of the domain or a potential of the domain. An a priori belief on the conditioning domain is allowed.

The function generates a new belief object containing the potential generated by the DRC, with names BSHN, BFLN.

DRC
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
CNDB = bshn
the belief short_names where the conditional belief is stored.
PRBT announces that what follows concerns the conditioning domain T
BSHN bel_shn / SET mode focal_set_description
PRBX (opt) announces that what follows concerns the domain X
BSHN bel_shn / SET mode focal_set_description
COMP announces that computation is to be performed.

The instructions BSHN bel_shn / SET mode focal_set_description can be one of:

- BSHN bel_shn provides the short_name of the belief object that contains either the a priori belief on T (case PRBT) or the belief about a dubious observation (case PRBX)
- SET mode focal_set_description where mode is one of B, O, D, X and focal_set_description is the binary, octal, decimal, hexadecimal representation of the set on which conditioning (either on T or on X) is performed.

User does not have to enter PRBX data. He must provide the PRBT data.
LIMITATION: the domain of bshn may contain only one variable:
(|belief(.).domain_variable| = 1).

5.5.5 CCR

To execute the CCR algorithm, all you need is a list of belief objects. They do not have to be defined on the same domain. Each belief can be discounted before performing the combination. It generates a new belief object containing the potential generated by the GBT, with names BSHN, BFLN. Coefficient disc = 0 means no discounting, disc = 1 means full discounting, the result being a vacuous potential.

The DISC instruction is necessary and every coefficient must be entered (use 0 if you don't want to discount). There must be as many values as there are belief functions involved.

CCR
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
BELF = bshn1,bshn2,...,bshnN
 a list of belief short_names, those to be included in the CCR.
DISC = x1,x2,...,xN
 the discounting factors, a list of reals in [0,1], one for each variable in domain_variable

5.5.6 DCR

To execute the CCR algorithm, all you need is a list of belief objects. They do not have to be defined on the same domain. Each belief can be discounted before performing the combination. It generates a new belief object containing the potential generated by the GBT, with names BSHN, BFLN. Coefficient disc = 0 means no discounting, disc = 1 means full discounting, the result being a vacuous potential.

The DISC instruction is necessary and every coefficient must be entered (use 0 if you don't want to discount). There must be as many values as there are belief functions involved.

DCR
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
BELF = bshn1,bshn2,...,bshnN
 a list of belief short_names, those to be included in the CCR.
DISC = x1,x2,...,xN
 the discounting factors, a list of reals in [0,1], one for each variable in domain_variable

5.5.7 CAUTIOUSCC

You enter a short (and a full) name for the belief object that will store the result of the cautious combination and the list of belief potentials to be combined.

CAUTIOUSCC
BSHN = a_short_name
BFLN = a_full_name, = to BSHN is absent.
BELF = bshn1,bshn2,...,bshnN
a list of belief short_names, those to be included in the CAUTIOUSCC.

5.5.8 GBTPDF

A special purpose function used to execute a GBT based on the belief on X given T, when the belief on the X variable is given by a probability density function (pdf), the a priori on T is vacuous, and the observation is a point or an interval in X. X is based on a frame which type is Continuous with domain $(-\infty, \infty)$.

The pdf can be understood as the Bayesian belief function that describes the user's beliefs or the pignistic probabilities that the user would use to bet on the value of X. In the second case, we generate the underlying q-least committed (q-LC) belief function isopignistic with the given pdf. (It means: we build the set of all the belief functions whose pignistic probability function is the given pdf (isopignistic). Among them we select the one that has the largest value for every commonality (q-LC).)

This program only compute Bel, BetP and Pl on the elements of their frame T. Its purpose was to show that TBMLAB can cope with a T domain with 100 elements, and BetP can still efficiently be computed (in less than 1 sec).

GBTPDF
BSHN = a_short_name, for the belief object that will store the results
BFLN = a_full_name, as above, = to BSHN is absent.
CNDB = bshn
the belief short_name that stores the conditional belief function
DATA = x1,x2 if the observation is an interval
= x1 if the observation is a point

For the conditional belief function, its potential must all be of the same type. Belief objects must obey some rules:

For each element of the T domain, you must define one POTENTIAL. POTENTIAL uses the PRDF option. PRDF uses one line of instruction. Next instruction is PEND. The instruction line is 'pdf_nature option1 option2 x_1 x_2' where

- pdf_nature is one of {gaussian, laplace},
- option1 is one of {bbf, Betf}. bbf means the entered pdf represents the user's beliefs. Betf means the pdf is the pignistic transformation of the user's underlying beliefs, and we know nothing else about these underlying beliefs.
- option2 is one of {ab, ms}. It means that the parameters entered are the a,b parameters or the mean, standard deviation parameters.

- x_1 and x_2 are two reals (the values of the 2 parameters announces by option2

The GBT is executed by the DATA instruction. Results are displayed on the screen. Batch Mode is interrupted by the display. You quit the display with the File/Quit menu, and come back in the batch mode environment in order to execute the instruction given by the next line.

5.5.9 RUN

It can be very convenient to be able to use a MATLAB function in batch mode. Of course it requires the knowledge of MATLAB and of TBMLAB programming languages.

Whatever is in the line that follows RUN is evaluated by an eval function. So the instructions:
RUN
generate_100_pdf('shifted')
become eval('generate_100_pdf(' 'shifted' ')').

5.5.10 PRINT

You can ask that the content of the belief object and its potentials be printed on the Matlab Command Window (case where FILE is absent) or on a file which name is the name on the FILE instruction.

You can ask for one belief object which name is given b y the BELF instruction. You can also put 'all' as in BELF = all. In that case, all belief objects are printed (on window or file, as asked for).

For every potential, you get a list with the masses and a second list with the triple Bel, BetP, Pl on every elements.

PRINT
FILE = file_name (optional)
BELF= bshn

5.5.11 REPORT

The REPORT provides the list of all objects in TBMLAB. Printed on the MATLA Command Window.

Chapter 6

Reference Manual

Contextual help. Many windows have their own contextual help buttons. By selecting them, you can obtain help focused on the object on which you work. You can also ask for activating the button attached helps, i.e., help texts explaining what the button under the cursor is doing. Tooltips are also available.

The Windows. TBMLAB manipulates many windows. A missing window would crash it. So to avoid inadequate window manipulations, user may NOT close them. The Window Close button (the □ on Mac and the Cross-in-Box button on PC) are desactivated. Reseizing windows is also desactivated. All you can do is to move them around (a bad idea).

Quit. You should always use the quit option when you are through and you want to return to the main menu window (the Root Window).

Crash. When system crashes, windows are usually inaccessible, so use the Matlab instruction: 'close all force' in order to close all the TBMLAB windows. One or two CTRL+C might be needed to access the MATLAB Command Window when you crash. If the CTRL+C does not act, select another window of Matlab or TBMLAB (like the Root Window), and it usually works. You can also move windows around so to make the MATLAB Command Window visible. Select it and proceed with CTRL+C.

When you crash and you have used tbmlab(0) to launch TBMLAB (the option with debug trace), you can only close the windows with the instruction 'close force all' typed in the MATLAB Command Window.

Under UNIX, when TBMLAB is hanging, go to the terminal (hit CTRL+Z to get a command prompt if necessary), then type 'ps' to see the pid number of Matlab, say it is xxx, you can type 'kill -9 xxx'.

6.1 The objects

A belief potential quantifies the belief held by an agent at a given time about the actual value ω_0 of a variable given all what the agent knows at time t (called the evidential corpus).

The full notation is:

$$\beta^{\text{domain}}_{\text{Agent,time}}\{\text{index}\}[EC](\omega_0 \in \text{domain-subset})$$

like in

$$\beta^{\Omega}_{You,t}\{\text{John}\}[EC](\text{age} \in [\text{20-40}]),$$

which means: the potential (β) held by agent You at time t that the age of $John$, which possible values of those in Ω, is between 20 and 40, given $Your$ evidential corpus at time t is EC.

The domain is the set of elements on which beliefs are assessed. In the finite case β is defined on the elements of the power set of Ω. In the continuous case, β is defined on the elements of the Borel sigma-algebra of intervals generated by the set of reals.

6.1.1 The belief potentials

Beliefs can be mathematically represented under many forms which are in one-to-one correspondence. They are the :

- m : the basic belief assignment (bba)
- b : the implicability function
- bel : the belief function
- pl : the plausibility function
- q : the commonality function
- w_c : the conjunctive canonical weights
- w_d : the disjunctive canonical weights

We use the name belief potential for any of these functions and denote it by β.

We also use the following convention for their normalized forms (when $m(\emptyset) = 0$).

- M : the normalized bba
- Bel: the normalized belief function
- Pl : the normalized plausibility function
- Q : the normalized commonality function

A belief potential expresses the belief holder opinions at time t given his/her evidential corpus, about the actual value of a **variable**. 'The Income of John in £' and 'The Income of John in \$' are two variables, Income is an **attribute**, John is the **index** of the attribute, and £ and \$ are **frames**.

So a **variable** ('the Income of John in £') is made of an attribute (income), a frame that expresses its possible values (£), and the instantiation of the attribute index (John).

The **attribute** is made of a structure (value_of_income), an index (someone) which can take value in the frame Person which contains John as an element.

The **structure** (value_of_income) is a set of frames (£ and \$), and a set of mappings.

The **mapping** expresses the relation between the frames of a same structure (1 £= 1.25 \$).

The **frames** are lists of elements (£, \$, Person).

A **belief potential** is defined on a frame of discernment Ω which is made of a set of elements, also called hypotheses, alternatives, worlds. The elements of Ω are the elements of the cartesian product of a set of variables (this set may be so degenerated that it contains only one variable).

6.1.2 The objects

TBMLAB builds several kinds of objects:

- frame
- structure
- mapping
- attribute
- variable
- belief

Each object has

- a number, assigned by TBMLAB, and that cannot be changed. It is the number used in TBMLAB to identify the objects.
- a short name that should really be short (2-4 characters). Same characters as the full name, but space and _ are forbidden. They must start with a letter.
- a full name that can be any string of alphanumeric symbols, or space or _. They must start with a letter. By default full names are set equal to the short names.

User does not have to know the detailed structure of the objects. TBMLAB wizard helps user all the time.

Other fields are explained in their individual description.

Tables 6.1.2 and 6.1.2 present the list of the fields for each object. Detailed explanations are given in the following sections dealing with the individual objects.

The initial default values of each field is listed in the file Function_Lib/Fcn_Utilities/Object_Utilities/Handling_objects/make_empty.m. This information is supposedly reserved to the developers.

Many information about the objects and their fields are listed in the file Function_Lib/For_Developers/structure_object.m. These are supposedly reserved to the developers.

frame	structure	mapping	attribute	variable
number short_name full_name type cardinality elements min max	number short_name full_name most_refined sub_frames mapping		number short_name full_name structure index(i).name index(i).frame	number short_name full_name attribute frame index_value

Table 6.1: Frame, structure, mapping, attribute and variable objects.

belief	belief().bp	belief().bp().form
number short_name full_name domain_variable domain_cardinality conditioning_variable conditioning_cardinality agent time ec origin computed disposable dci_var.pin_d dci_var.pin_c dci_var.pin_i	number full_name cd_focal_set condition form	type focal_sets values representation log_expr contint

Table 6.2: Belief object and its sub-objects.

6.1.3 The frame object

A frame is essentially a set of elements.

Example 6.1.1. Imagine the following frames. We list their short name, full name, type and elements.

ShN	full names	type	elements
Age1	age in years	Integers	min 0 max 119
Age2	age in decades	Classes_Ordered	[0-9], [10-19], ... [110-119]
Age3	age in classes	Classes_Ordered	[0-25], [26-60], [61-119]
Age4	young_age in years	Integers	min 0 max 25
Prs	Individual Names	Classes_Un_ordered	All, Brown, Carol, ...
Mar	Being_Married	Logical	False, True
Yr	Years	Integers	min 1995 max 2004
Car	Owned_Car	Classes_Un_ordered	Audi, BMW, ...
Inc	Income	Classes_Ordered	Low, Medium, High

□

The fields of the object frame(i):i=1,2 . . . , are:

- frame(i).number : the number given by TBMLAB
- frame(i).short_name : the short name
- frame(i).full_name : the full name
- frame(i).cardinality : a number, its cardinality (number of elements). Skip if type = Continuous or Integers.
- frame(i).elements : the list of elements. Elements are strings of characters. Skip if type = Continuous or Integers
- frame(i).type : the type of frame can be any of
  - Logical
  - Binary
  - Classes_Un_ordered
  - Classes_Ordered
  - Integers
  - Continuous
- frame(i).min : the minimal value, only for integers and continuous
- frame(i).max : the maximal value, only for integers and continuous

For continuous frames, the cardinality is ‘infinity’.

When min or max are infinity or -infinity, you enter the text infinity or infinite or inf preceded by - if negative. They are synonymous.

Infinity is numerically equal to 1E40 (by convention).

For Continuous frames, the elements field is empty.

For Integers frames, the elements are integers from min to max.

Element labels are stored with the MATLAB strvcat instruction.

6.1.4 The structure object

Several frames can concern the same attribute. For instance, consider the age_of_John. It can be expressed on the frame Age1, Age2, Age, and Age4. These four frames must be built as frames, and are regrouped into one structure. So a structure is a set of frames. All frames in a structured are a coarsening or an inclusion of some most refined frame. The other frames are called sub_frames.

Example 6.1.2. Imagine the following structures. We list their short name, full name and frames.

ShN	full name	frames
Sag	Age	Age1, Age2, Age3, Age4
SPrs	Person_Str	Prs
SMr	Married_Str	Mar
SYr	Years_Str	Yr
SCar	Car_Str	Car
SInc	Income_Str	Inc

□

The fields of structure(i),i=1,2, . . . , are:

- structure(i).number
- structure(i).short_name
- structure(i).full_name
- structure(i).most_refined : the number of the most refined frame
- structure(i).sub_frames : the list of the numbers of the other frames
- structure(i).mapping : a matrix which element c(i,j) is the number of the mapping object that expresses the relation between frame i and frame j.

6.1.5 The mapping object

Age2 and Age3 are coarsenings of Age1. Age4 is an inclusion of Age1.

Mapping provides the relations between these frames. NA

6.1.6 The attribute object

An attribute is a matter of interest characterized by a structure and possibly several indexes.

Example 6.1.3. Imagine the following attributes. We list their short name, full name, structure, index names and frames. The fourth attribute has two indexes, the second has none.

ShN	full name	structure	Index	frame
Age	Age(Ind)	SAge	Individual	Prs
Rain	It_rains	Logical		
Car	Owned_Car(Ind)	SCar	Individual	Prs
Inc	Income(Ind,Year)	SIn	Individual	Prs
			Year	Yr

□

The fields of attribute(i),i=1,2, . . . , are:i

- attribute(i).number
- attribute(i).short_name
- attribute(i).full_name
- attribute(i).structure : the number of its structure object
- attribute(i).index(1).name : the name of the first index
- attribute(i).index(1).frame : the number of the frame that lists all the possible values for the first index
- attribute(i).index(2).name : the name of the second index

- attribute(i).index(2).frame : the number of the frame that lists all the possible values for the second index
- attribute(i).index(3).name : etc . . . .

The field index can be absent when there are no indexes.

6.1.7 The variable object

A variable is an attribute defined on a specific frame (one of those in the structure of the attribute), and an instantiation of every index of the attribute. Beliefs concern the actual value of a variable.

Example 6.1.4. Imagine the following attributes. We list their short name, full name, frame, index instantiations.

ShN	full name	frame	index instantiation
Age(A)	Age(Allen)decades	Age2	Allen
Car(C)	Owned_Car(Carl)	Car	Carl
Inc(B,99)	Income(Brown,1999)	Income_Str	3,8

□

The fields of variable(i),i=1,2, . . . , are:

- variable(i).number
- variable(i).short_name
- variable(i).full_name
- variable(i).attribute : the number of its attribute object
- variable(i).frame : the number of its frame object
- variable(i).index_value : the value of each index. The value is the position of the element in the element list of the frame related to this index and defined in the attribute object.

Note. The generation of variables is highly simplified thanks to this architecture. Once an attribute has been defined with several indexes, you can ask that TBMLAB builds all the possible variables that can be built by using all possible valued of the index. The only constraint is that all these variables use the same frame.

So if the attribute income has two indexes, person and year, person frame is People1 to People8, and year is integers between 1996 and 2005, you go to the variable query list and ask in one instruction to get the 80 variables at once. They will be income(people1,1996), income(people1,1997), . . . income(people1,2005), income(people2,1996), . . . income(people8,2005).

Similarly consider position (target,time) with five different targets and several times . . . Creating each variable one per one is very tedious. TBMLAB does it for you.

6.1.8 The belief object

The belief object is the object that stores the belief potentials defined on a set of variables.

If the belief is a marginal belief (unconditional), there is one belief potential to store. When you create a marginal belief, TBMLAB generates one vacuous bba.

If the belief is a conditional belief, there are as many belief potentials to store as there are elements in the conditioning domain. When you create a conditional belief, TBMLAB generates a vacuous bba for every element of its conditioning domain.

To enter the actual potential, you use the edit option in the Belief Editor Window. Unedited potentials remain vacuous bba's. Each potential can be stored under different forms, like a *m* and a *bel* function.

The fields of belief(i),i=1,2, . . . , are:

- belief(i).number
- belief(i).short_name
- belief(i).full_name
- belief(i).disposable : technical, states if belief object can de deleted by garbage collector (see menu Edit/Clear unused beliefs)
- belief(i).agent : the name of the belief holder (a string)
- belief(i).time : the time at which this belief is held (a number)
- belief(i).ec : a free text stating what knowledge is accepted by the agent when building the belief.
- belief(i).domain_variable : the list of the numbers of the variables for the domain of the belief potential
- belief(i).domain_cardinality : technical, the cardinality of every variable in the domain of the belief potential
- belief(i).conditioning_variable : the list of the numbers of the variables for the conditioning domain, only discrete variables.
- belief(i).conditioning_cardinality : technical, the cardinality of every variable in the conditioning domain.
- belief(i).origin : technical, the numbers of the used belief potentials and the indexes of the used forms of these belief potentials.
- belief(i).computed : technical, a text on how the belief was computed
- belief(i).dci_var.pin_d : matrix I×J, line i = [position_i, index_i, number_i] for i-th discrete domain_variable (Discrete variables are all the variables that are neither continuous nor integers.).
- belief(i).dci_var.pin_c : matrix I×J, line i = [position_i, index_i, number_i] for i-th involved Continuous domain_variable.

- belief(i).dci_var.pin_i : matrix I×J, line i = [position$_i$, index$_i$, number$_i$] for i-th involved Integers domain_variable.
- belief(i).bp(j) : the j'th belief potential sub-object, one for each conditioning domain element (one for marginal belief potentials).
  - belief(i).bp(j).number : the number allocated to this potential.
  - belief(i).bp(j).full_name : the name allocated to this belief made of its number and the types of forms available (example 'bp43 m b BetP').
  - belief(1).bp(1).condition : a character string detailing the conditioning element (example : 'age_category=4,sex=Male').
  - belief(i).bp(j).cd_focal_set : the position of the element in the conditioning domain to which bp(j) corresponds. As one bp exists for every element, its value is normally j.
  - belief(i).bp(j).form(k) : the k'th form sub-sub-object, one for each stored form of belief potential i.
    - belief(i).bp(j).form(k).type : the type of this form, any of {m, b, bel, pl, q, wc, wd, M, Bel, Pl, Q, BetP, m_pdf, BetP_pdf, m_from_BetP_pdf};
    - belief(i).bp(j).form(k).representation : 'selected' when values are stored only for some selected subset of Ω, 'ordered' when values are stored for every subsets of Ω (in that case the order of the values is essential), or 'beta' or 'gamma' or 'gaussian' or 'laplace', or 'beta_based' or 'gamma_based' or 'gaussian_based' or 'laplace_based'.
    - belief(i).bp(j).form(k).focal_sets : list of the subsets considered in the 'selected' representation case
    - belief(i).bp(j).form(k).values : values of the belief potential stored in the same order as the focal sets in the 'selected' case, or the two parameters for the pdf.
    - belief(i).bp(j).form(k).log_expr : a cell array with the logical expression describing the focal sets (when logical expressions have been used to generate the focal set)
    - belief(i).bp(j).form(k).contint(n).limits : the matrix of the limits of the intervals (only for continuous and integers variables) related to the value n, thus related to belief(i).bp(j).form(k).values(n). See below.

Note: Mixing Discrete variables with Continuous and Integers variables. The variables in belief(i).domain_variable are separated into two groups, the discrete variable group that contains all the variables except the continuous and integers ones, and the continuous and integers group that contains the continuous and integer variables.

For the second group there are strong limitations. What can be represented? As an example, consider two discrete variables D$_1$ and D$_2$, one integer variable Int and one continuous variable Cont. Suppose the elements of D$_1$ are x_1 and x_2, and those of D$_2$ are y_1, y_2 and y_3. The limits of Int are 5 and 20. The limits of Cont are 10.5 and 30.8. There are thus 6 elements in the discrete variable domain.

For Integers or Continuous variables $V_1, V_2, ...V_N$, and $S_i \subseteq V_i : i = 1, ..., N$, the set $R = \{v_1, v_2, ..., v_N : v_i \in S_i, i = 1, ..., N\}$ is called a rectangle.

The universe is represented as shown in table 6.3

case	D$_2$	D$_1$	Int		Cont	
1	y_1	x_1	5	20	10.5	30.8
2	y_1	x_2	5	20	10.5	30.8
3	y_2	x_1	5	20	10.5	30.8
4	y_2	x_2	5	20	10.5	30.8
5	y_3	x_1	5	20	10.5	30.8
6	y_3	x_2	5	20	10.5	30.8

Table 6.3: The universe.

If you select the six lines, you get the representation of the universe.

For another subset, you select those cases that belong to it, and you can change the limits of Int and Cont. Table 6.4 is a subset with 3 discrete elements, and each one has its own 'rectangle'. It contains case 2, 3 and 5. The rectangle attached to case 2 (y_1,x_2) is Int $\in [10, 12]$ and Cont $\in [10.5, 30.8]$. The rectangle attached to case 3 (y_2,x_1) is Int $\in [15, 15]$ and Cont $\in [16, 18.3]$. The rectangle attached to case 5 (y_3,x_1) is Int $\in [5, 20]$ and Cont $\in [10.5, 30.8]$. Hence a subset admits one rectangle for each discrete element. This is the limitation about the Continuous and Integers variables.

case	D$_2$	D$_1$	Int		Cont		Selected
1	y_1	x_1	5	20	10.5	30.8	N
2	y_1	x_2	10	12	10.5	30.8	Y
3	y_2	x_1	15	15	16	18.3	Y
4	y_2	x_2	5	20	10.5	30.8	N
5	y_3	x_1	5	20	10.5	30.8	Y
6	y_3	x_2	5	20	10.5	30.8	N

Table 6.4: A subset.

For what concerns the the Continuous and Integers variables, we store a matrix with the case number (so we know the elements it is related to), and the pairs of limits. If all pairs of limits in one line are those of the universe, the line is omitted. Hence the matrix representing the universe is empty. Table 6.5 is the matrix stored in the limits field.

case	Int		Cont	
2	10	12	10.5	30.8
3	15	15	16	18.3

Table 6.5: Stored matrix.

More formally, consider potential bp(j). Consider the mass m(n) given to the n'th focal sets : m(n) = belief(i).bp(j).form(k).values(n). Its focal set is made of two parts.

- for the discrete variable group belief(i).bp(j).form(k).focal_sets(n,:) is the long number that represents the subset of the discrete variable domain related to m(n).
- for the continuous and integers group, belief(i).bp(j).form(k).contint(n).limits contains the limit matrix.

In the full notation, the fields become:

$$\beta_{\text{Agent,time}}^{\text{domain\_variable}}\{\text{ind}\}[\text{ ec,value of conditioning\_variable }] \tag{6.1}$$

where ind is the value of the indexes of the attributes of the used variables.

Further details about the belief potential are presented in section 6.8.

Note: Domain variable made of only one Continuous or Integers variable. For unidimensional potentials, we can handle a few special cases when the variable is Continuous or Integers. We can enter a probability density function (pdf).

- form().type = m_pdf : pdf values are those of the bba.
  form().representation = one of {binomial, beta, gamma, laplace, gaussian}.
  form().values = one of { [p,n], [a, b, μ, σ], [t, n, μ, σ], [$\alpha, \beta, \mu, \sigma$], [$\mu, \sigma$]} (same order as representation options)
- form().type = BetP_pdf : pdf is the pignistic probability function of an underlying bba.
  form().representation = one of {binomial, beta, gamma, laplace, gaussian}.
  form().values = one of { [p,n], [a, b, μ, σ], [t, n, μ, σ], [$\alpha, \beta, \mu, \sigma$], [$\mu, \sigma$]} (same order as representation options)
- form().type = m_from_BetP_pdf = m to be computed as the q-Least Committed bba isopignistic with the pdf stored in the BetP_pdf form.
  form().representation = one of {binomial_based, beta_based, gamma_based, laplace_based, gaussian_based}.
  form().values = one of { [p,n], [a, b, μ, σ], [t, n, μ, σ], [$\alpha, \beta, \mu, \sigma$], [$\mu, \sigma$]} (same order as representation options)

Normally the second and third cases go together. The explanations of the distributions and of their symbols are in Abramowitz and Stegun, Handbook of Mathematical Functions, Dover, section Probability Functions (Abramowitz & Stegun, 1965).

For the binomial pdf (p,n), the domain must be Integers with min = i, max = i+n. Classically i = 0.

For instance, we can have:

- case Integers and Bayesian belief function (the pdf is the bba)
  belief(ib).bp(ip).form(it).type = 'm_pdf';
  belief(ib).bp(ip).form(it).representation = 'binomial';
  belief(ib).bp(ip).form(it).values = [p n];
- case Integers and BetP
  belief(ib).bp(ip).form(it).type = 'BetP_pdf';

belief(ib).bp(ip).form(it).representation = 'binomial';
belief(ib).bp(ip).form(it).values = [p n];
belief(ib).bp(ip).form(is).type = 'm_from_BetP_pdf';
belief(ib).bp(ip).form(is).representation = 'binomial_based';
belief(ib).bp(ip).form(is).values = [p n];

- case Continuous and Bayesian belief function (the pdf is the bba)
  belief(ib).bp(ip).form(it).type = 'm_pdf';
  belief(ib).bp(ip).form(it).representation = 'beta' or 'gamma' or 'gaussian' or 'laplace';
  belief(ib).bp(ip).form(it).values = [a b μ σ] or [t n μ σ] or [μ σ] or [α β μ σ];

- case Continuous and BetP
  belief(ib).bp(ip).form(it).type = 'BetP_pdf';
  belief(ib).bp(ip).form(it).representation = 'beta' or 'gamma' or 'gaussian' or 'laplace';
  belief(ib).bp(ip).form(it).values = [a b] or [t 1] or [mu sigma];
  belief(ib).bp(ip).form(is).type = 'm_from_BetP_pdf';
  belief(ib).bp(ip).form(is).representation = 'beta_based' or 'gamma_based' or 'gaussian_based' or 'laplace_based';
  belief(ib).bp(ip).form(is).values = [a b μ σ] or [t n μ σ] or [μ σ] or [α β μ σ];

Beware that some distributions are not yet fully implemented. For the moment, they are used only within the GBT as data distributions. Combination and conditioning not yet implemented.

When there are four parameters listed, you must only enter one pair of parameters (first and second, or third and fourth), the other pair in automatically computed.

Note. If you want to build the same belief on sets of variables that only differ by their indexes instantiation, it can be done very efficiently. You will build one such belief for a given instantiation, and then ask TBMLAB to reproduce it for every possible values of the indexes. It fits with the next example. Let three attribute $X \in \mathcal{X}$, $Y \in \mathcal{Y}$ and $Z \in \mathcal{Z}$ where X depends on index i, Y depends on indexes i and j, and Z depends on indexes j and k. So they could be written as $X(i), Y(i,j), Z(j,k)$. We then take every possible combination of i, j and k. For each such assignment, TBMLAB defines a belief object. It corresponds to

$$\forall i \in I, \forall j \in J, \forall k \in K,$$

$$bel(X(i) \in A, Y(i,j) \in B, Z(j,k) \in C) = f(A,B,C)$$

$$\forall A \subseteq \mathcal{X}, B \subseteq \mathcal{Y}, C \subseteq \mathcal{Z}$$

where $f(A,B,C)$ does not depend on i,j,k. The indexes i,j,k correspond to the information ind in {ind} is relation 6.1

6.2 The File menu

The options available in the File menu are:

1. Open : open a file with previously saved binary data or with batch mode instructions
2. Save : saving data
3. Save as : saving data on a new file
4. Print
5. Quit

Details of each option are given in the next subsections.

6.2.1 Open

You can choose between loading previously saved objects, or creating the objects in batch mode using a coded form.

When selecting the Open menu, you receive the system panel for opening a file. Just select the folder and the file. Normally, files with .batch are batch mode files, those with .obj.mat are binary files, those with .EVN are EVN data. These three suffixes are needed by TBMLAB. It uses them to know how to process the data. Other suffixes exist like .belprint, but they are unnecessary.

Once Open menu has been used, it cannot be reused during the same session. TBMLAB would be lost with objects found in both files... Hence to start a new session, you must quit TBMLAB (not MATLAB) and relaunch it.

Case : Binary Files thus .obj files. TBMLAB loads the data you have saved from a previous session. Loaded data are the frame, structure, mapping, attribute, variable, belief objects.

Case : Batch Mode thus .batch files. Instructions for building all the objects are entered with a form created with any text editor. Instruction obeys to a special form which syntax is detailed in section 3.6.

6.2.2 Save

Save on file all objects in binary form. Data can be recovered with the File/Open menu.

6.2.3 Print Report

Print a report on MATLAB Command window or in an ascii file.

6.2.4 Quit

The official way to leave TBMLAB and go back to MATLAB. Every window is closed, and you are back to MATLAB Command Window No file is saved during this process, so you are asked to confirm your request. If you want to save data, just cancel the option, save your data and repeat the request then.

6.3 The Edit menu

The options available in the Edit menu are:

1. Variables
2. Short Cuts
    (a) Natural frames : building some classical frames
    (b) Logical propositions : building variables that are logical propositions
3. Mapping
4. Belief
5. Data Verification
6. Clear unused beliefs

Details of each option are given in the next subsections.

6.3.1 Variables

The editor for the objects frame, structure, attribute, variable is entered with menu Edit/Variables.

You receive a display (Object Editor window) with four columns and a panel in the upper left corner.

The panel reminds what the objects are and what action must be performed.

Each column has a label (Frame, Structure, Attribute, Variable). It lists the available objects.

To View or Edit an existing object, select the list that contains it (note that with Unix, you must click on the displayed objects, not just in the list window). If you double click fast enough on an object, you get the data of the selected object. Else you get a panel , you select the object and select Done. At lower left corner, you get a panel with the fields of the object. You can edit its content. You must also select the object required by the one you are creating. So to create an attribute without index, you must select a structure. Other panels may show up when needed, like to build the elements, the indexes... When the objects and the fields are as you want, select the OK (red) button. Your lists are updated.

To create a new object of type xxx,

- if the xxx list is empty, just type in it...
- if the xxx list is not empty, select menu Edit/New xxx
- if the xxx list is not empty, just type in it and select New in the panel that pops up.

Then proceed as with edit. The Edit menu concerns also the deletion of objects. You select the type of object to delete, then you go in their list, multi-select those to be deleted, you select the red button, you confirm your choice, ... and all these objects disappear, as well as all those that use them, or use an object that use them...

When you delete an object, TBMLAB deletes every object which meaning depends on the deleted object. For instance, if frame 3 is deleted, the structure with most_refined = frame 3 is deleted, as well as the attribute which value is based on the deleted structure or which index is based on the deleted frame, the variable which depends on the frame 3 or the deleted attribute, and the belief which used a deleted variable. TBMLAB helps you in showing all objects that are going to be deleted if you confirm your order. All are then deleted simultaneously. So don't be surprise that a single delete induces many deletions.

Beware, delete can be devastating, better save your data before performing such an action. It cannot be canceled.

6.3.2 Natural frames

An easy way to generate frames that cover most real life situations. In Get Natural Frames window, you get a list of ready to use classical frames. Their properties (names, cardinality, elements names) are provided by TBMLAB.

1. Logical False True
2. Binary 0-1
3. Binary No-Yes
4. Binary Male-Female
5. Binary Friend-Foe
6. Binary + -
7. Binary For-Against
8. 2-ary A-B
9. 3-ary A-B-C
10. 4-ary A-B-C-D
11. *k-ary A-B-C-D ...
12. 2-ary ordered 1-2
13. 3-ary ordered 1-2-3
14. 4-ary ordered 1-2-3-4
15. *k-ary ordered 1-2-3-4 ...
16. *Integers from-to NA
17. *Continuous from-to NA
18. All of them except those preceded with *
19. Do variable for each selected frame.

To get any of these frame, just click on the corresponding button located at their left. Unselection is obtained by clicking again the button. Once a frame exists, its button disappears. Multi-select those you want.

If you select the option 'All of them except those preceded with *' , you get most of them at once (those without *). When there is a *, you are asked further questions (the k values and the limits).

If you select the option 'Do variable for each selected frame.', TBMLAB builds for every selected frame a structure with one frame, an attribute without indexes and which structure is the previous one, a variable built on the previous attribute and on the initial frame. It is a convenient short cut to get at once frames, structures, unindexed attributes and variables.

6.3.3 Logical propositions

If you want to use logical propositions, use this option. You will have to enter their number and their names. If the logical frame and structure did not exist, they are created.

You get queries in the Get Propositions window which must be accepted by clicking the OK red button. The whole process can be stopped at any moment by clicking the cancel red button. The queries are :

1. How many propositions you want to create now: give a number, click on the red button.

2. List of SHORT names for each proposition. Proposed default is pi, pi+1, ... pk where i in the first unused index for naming logical propositions. If there are too many propositions to fit in one page, when hitting the red button you get next part. You can feed new short names for every proposition, like p, b, f, etc ... by erasing the default and typing your choice. Proposition short names must be composed of letters, digits and _, nothing else, no blank, and starting with a letter. Use really short names (a few characters, 1, 2, 3 ...), it helps, but is not necessary (you will just have to type more characters and displays will be more clumsy).

3. When short names are over, you are request to enter their long names. The default are the short names. If you are happy so, click on the OK red button, otherwise correct those you want to change. To get next list, click on the red button. When over click on the red button, you are back to the Root Window.

The result of this operation is that frame, structure, attribute and variable objects have been created.

Comments. Think about having 3 propositions like in the DBG diabetes example. You input 3 for proposition number, and d, b and g as short names. This is all you have to do. No need to write things like $\neg$ or $\sim$, etc ...

6.3.4 Mapping

NA

6.3.5 Belief

The object that stores the belief functions (or any of its transformations) relative to a set of variables. Made of several belief potentials if beliefs are conditional or one belief potential if belief is not conditional.

This menu allows you to edit one belief object.

Beware that all variables needed to build a belief object must have been defined before entering the belief object. Otherwise back up (quit menu) and create them.

You get a list of all existing belief objects (number and full name) and a list of all existing variable objects (number and full name).

You cycle among the next options presented in the panel till you choose quit.

1. new : you create a new belief object.
2. edit belief: you can display an existing belief object that you select in the list at right.
3. edit bba : you only edit the bba of a belief object, thus you skip a few useless panels.
4. delete : you delete a belief object.
5. Duplicate → index : duplicate one belief for all possible values of the indexes of its variables.

New

1. You receive a list of queries which answer must be typed in. Answer them. Click red button.
2. It asks which variables should be used for the domain of that belief potential. In the variable list, you select one or several variables. The domain (frame of discernment) of the belief potential under construction is the product space made with these variables.

   Multi selection is done by clicking on the variable number/name field while holding down the Command key (Mac) or Control key (PCWIND).

   When selection done, click red button.
3. It does the same for conditioning variables. Beware, no variable can be both a domain variable and a conditioning variable.
4. It ask you if you want to enter into the belief potential editor. Click on the Yes red button.

You enter into a window called the ‘Potential Control for belief number k’, where k is the belief object number.

For details on the belief potential editor, see section 6.8.

6.3.6 Data Verification

TBMLAB checks that the objects are present and syntactically correct.

6.3.7 Clear unused beliefs

Belief objects that are no more necessary are deleted from memory. Useful to get memory space back in some computer environments.

6.4 The View menu

The options available in the View menu are:

1. All objects
   (a) Objects on Cmd_Window
   (b) Objects in file
2. All object fields
   (a) Fields on Cmd_Window
   (b) Fields in file
3. One object fields
   (a) Fields on Cmd_Window
   (b) Fields in file
4. Object Road Map
5. Hide Windows

Details of each option are given in the next subsections.

6.4.1 All objects

All objects are listed with their values. Results are displayed on the MATLAB Command Window or are saved on a file and can be loaded by text editors.

6.4.2 All object fields

Show object fields, hardly readable,very long, but may be useful for debugging.

6.4.3 One object fields

One object fields are listed on the MATLAB Command Window or on a file and can be loaded by text editors.

6.4.4 Object Road Map

A full screen display that shows what objects make a belief. For every belief object, it provides the list of its variables. For each variable, it states what is the frame on which the variable is defined, the attribute object uses to define the variable, and the structure object of the attribute. Clicking on the little button to the left of the objects produces a window listing the data of the selected object.

6.4.5 Hide Windows

All windows are made invisible, but still exist. Useful to view and work on other windows. When hiding all the TBMLAB windows, you get a new little violet colored window. Move it around but keep it accessible as you will need it to get the windows back. Windows are brought back by just selecting this colored window.

6.5 The Tools menu

The options available in the Tools menu are:

1. Run a .m fct
2. Valuation Base Systems
3. Graphical Method
4. Distinct Combination
    (a) Distinct AND
    (b) Distinct OR
    (c) Distinct mixed
5. Cautious Combination
6. GBT
    (a) GBT:X/T both discrete
    (b) GBT:X Cont-Int,T discr
7. TBM Tabulator
8. New Forms/Frames
    (a) Get New Forms
    (b) Extension/Marginalization
    (c) Conditioning NA
    (d) Ballooning Ext NA
9. Approximate Distinct Combination NA
10. Compare 2 belief potentials NA
11. EvClus : dissimil → m
12. Preferences
    (a) Default Preferences
    (b) Open Preferences
    (c) Save Preferences
    (d) GUI_Parameters
    (e) TBM_Parameters
    (f) Help Level

Details of each option are given in the next subsections.

6.5.1 Run a .m fct

You can use a m.file that you have written and that builds the variables you need or run any user's function. Convenient to read data from other external data bases, or any data easier to generate by a special program. Convenient also for applications.

This function must build and/or use the various objects (like frame, attribute, variable, belief...) required by TBMLAB. When the function is implemented, you can call that function from here. You only enter its name (without the .m extension). Be sure its path is known by MATLAB (that it was in the Belief_Machine folder). TBMLAB will execute it using eval(the name of your function). The data are transferred into TBMLAB through the next global statement:

global frame structure mapping attribute variable belief.

Such a function can in fact perform whatever computation you want, but requires a serious knowledge of TBMLAB architecture and programs atop of a knowledge of MATLAB.

6.5.2 Valuation Base Systems (Dir/Undir)

The Valuation Base System is a propagation algorithm developed by P. Shenoy, and based on the Joint Binary Trees.

The Undirected form uses joint beliefs defined on the joint spaces made by several variables.

The Directed form use conditional beliefs defined on the joint spaces made by several variables, one for each element of the conditioning space. The Bayesian networks (the DAG) are particular cases of this option.

User introduces the list of belief objects to be combined, and the set of variables on which marginalization is requested.

In order to run VBS, TBMLAB needs to know which belief objects are involved in the network and on which subsets of variables the projection of the resulting propagated belief is to be computed.

As per now, it computes only one projection per request.

6.5.3 Graphical Method

The menu controls your requests about Evidential Network (EVN) and Conditional Evidential Network (CEVN). It can:

- display an EVN or CEVN,
- propagate beliefs in the EVN/CEVN,
- display BetP on every individual variables after propagation.

Normally every variable and prior belief should have been entered. An EVN/CEVN asks for a set of belief objects. It collects all involved variables. It draws the network, that you can move around, zoom in and out. You can also display the BetP on each single variable. BetP histogram is located above the node of its corresponding variable. When you move variables, the BetP histograms stay where they were. The menu Relocate BetP relocates the BetP histograms above their variables.

When entering the belief objects, you can enter all of them at once, but the plot might be very messy. So it it better to enter a first sets of belief objects with only a few of them, and then enter the other ones, one by one. There is on option (menu View/Nice Graph) that tries to build a nice network and works often nicely (algorithm provided by P. H. Wuillemin).

When working on the EVN/CEVN, you loose the access to the menus. A red button indicates it at upper left. To access the menus back, select the red button (Mac, click, other, right click), it becomes green and you can select the menus.

You can also add variables from the EVN/CEVN, and then create beliefs. It is NOT a good idea, better plan ahead.

Menus are attached to the nodes.

EVN/CEVN can be saved and reloaded.

6.5.4 Distinct Combination

Given several belief functions induced by distinct pieces of evidence, we combine them using :

- AND = conjunction operator corresponding to the conjunctive combination rule.
- OR = disjunction operator corresponding to the disjunctive combination rule
- Neg = negation operator.

Given two bba m_1 and m_2, their conjunctive combination is such that $m_1(A)m_2(B)$ is transferred to $A \cap B$. Given two bba m_1 and m_2, their disjunctive combination is such that $m_1(A)m_2(B)$ is transferred to $A \cup B$. For the bba m, its negation, denoted $\overline{m}$, is the bba where $\overline{m}(A) = m(\overline{A})$

Beliefs may be defined on different frames of discernment, vacuous extensions bring them back to a common frame.

Case 'Distinct AND'. Algorithm to perform the conjunctive combination of the beliefs induced by distinct sources.

Case 'Distinct OR'. Algorithm to perform the disjunctive combination of the beliefs induced by distinct sources.

Case 'Distinct mixed'. Algorithm to perform a mixture of conjunctive and disjunctive combinations of the beliefs induced by distinct sources. Negated beliefs are also accepted.

6.5.5 Cautious Combination

When the sources producing the beliefs are not distinct and their correlation is known, combination is immediate. When the sources producing the beliefs are not distinct and their correlation is not known (the usual case), we compute their most cautious combination. This combination rule is commutative, associative and idempotent. It works both for conjunctive and disjunctive combinations. Only the conjunctive form is implemented. It requires as input a potential with one of {m, M, wc, wd} form. It works only on marginal beliefs.

Results are saved in a belief object which short and full name are CC(sh1, sh2, ...) where sh1, sh2, ... are the short names of the beliefs used for the combination.

Maximal overall cardinality is 16.

6.5.6 GBT

Tools dealing with the General Bayesian Theorem (GBT).

GBT:X/T both discrete

You enter a potential with conditioning variables and a conditioning event. TBMLAB executes the GBT and produces a potential on the conditioning variable domain. Suppose the domain_variable domain is X and the conditioning_variable domain is Θ. You have one potential $m^X[\theta_i]$ on X for each $\theta_i \in \Theta$. The conditioning event is a subset x of X. The GBT produces a potential $m^\Theta[x]$ on Θ given x.

A first display lists all conditional belief objects available. Select one belief object to which you want to apply the GBT. At left, you receive two list, one with the domain variables, one with the conditioning variables.

To build the conditioning event, you will use the Bba Editor, and build a potential on X. For a certain event, build a bba with one focal set which receives a mass 1. Else enter a full belief on X as when conditioning on a dubious/noisy event. To do so, you get the Bba Editor window (see section 6.8.1, paragraph set mode). Build the conditioning subset x by selecting its elements. Enter the mass. Repeat and end with selecting menu File/Quit.

Do the same to build a potential on Θ.

TBMLAB does all the computation. It produces two potential:

- GBT_ConditEv that store the conditioning event $x \subseteq X$. It contains a selected m vector.
- GBT_xxx that stores the a potential on Θ given x, where xxx is the name of the the belief object with the potentials $m^X[\theta_i]$. It contains a selected m vector and a selected wc vector.

The result can be explored with the Edit/Belief menu.

Limitations: the cardinality of X and Θ must both be no larger than 18.

GBT:X Cont-Int,T discr

6.5.7 TBM Tabulator

A TBM oriented tabulator that performs on line computation on belief potentials. It is intended for computing potentials on one or two frames. Frame cardinality should be reasonably small (< 12).

You get a whole screen window containing a large tabulator, a header with helps, and a bottom window for control.

Hereafter:

- n is an integer representing the cardinality of the frame

- symb is any symbol, the labels of the elements of the frame are made of the ascii codes that follow symb.
- x,xi are cell locations: i,j (if x = j, then i = 1). Indices of the cell containing the generating instruction of a potential.
- out is a potential form.

In the tabulator, you type in instructions that are immediately executed. You can:

- create the labels of the frame of discernment with label(n;symb)
  ex: label(3;f) generates the labels , f, g, fg, h, fh, gh, fgh. fh denotes the set {f,h}.
- generate a random bba with bba(n)
  ex: bba(3) generates the bba for a frame with 3 elements.
- type in a potential with key(n;out)
  ex: key(3;pl) generates a vector of length 8 with 0's. You key in the values of pl. End by selecting outside object area.
- compute the various forms with op(x) where op is one of {m, b, q, bel, pl, wc, wd, betp}.
  ex: b(4), computes b from data under cell(1,4), q(3,5) computes q from data under cell(3,5),
- compute con/dis-junctive combinations of several potentials: ccr(x1; ... ; xn; out), dcr(x1; ... ; xn; out)
  ex: ccr(2;3;5,7;b) conjunctive combination of 3 potentials located under cells (1,2), (1,3), (5,7), output as b.
  ex: dcr(2;3;5,7;q) disjunctive combination of 3 potentials located under cells (1,2), (1,3), (5,7), output as q.
- apply GBT to several conditional potentials, with prior potential: gbt(x1; ... ; xn; xprior; conditioning_event; out)
  ex: gbt(3;5;8;11;4,9;m) applies GBT to conditional potentials in (1,3), (1,5), (1,8), prior in (1,11), conditioning event represented in cell (4,9), results as m.
- apply DRC to several conditional potentials, with prior potential: drc(x1; ... ; xn; xprior; out)
  ex: drc(3;5;8;11;m) applies DRC to conditional potentials in (1,3), (1,5), (1,8), prior in (1,11), results as m.
- discount a potential: disc(x;dc;out)
  ex: disc(5;.7;pl) discounts by factor .7 the potential in (1,5), results as pl.

The bottom window displays:

- A set of 4 buttons (Up, Down, Left, Right) allows you to scroll the tabulator.
- Home puts cell (1,1) at upper left of the tabulator.

- Clear erases all data, and gives back to you an empty tabulator.
- Replay allows to replay all the instructions. Useful if you use a key instruction and key in new data before running replay. Random bba"s are frozen or changed according to the option "Rep Fr" or "Rep Nw"

Menu

- Help : presents the help.
- Save : saves numerical data in a file named tbm_tabulator_data located in User_Data/Other_Data, ASCII, tab separated, readable by Excel...(so you can print/plot data).
- Quit : brings you back to TBMLAB Root Window.

6.5.8 New Forms/Frames

Get New Forms

Producing various forms of belief potential. When one form is stored and another one is requested, this task is performed by this menu. We can thus generate any of m, b, bel, pl, q, wc, wd from any other one, and store the result.

Avoid transforming wc and wd when one of their elements is 0, in which case m(universe) = 0 to m($\emptyset$) = 0, respectively. There is a 0 divider, and TBMLAB does its best and will avoid crash, but the results are not guaranteed.

Extension/Marginalization

Given a marginal belief potential define on space X, you can project it on a new space Y, where a space is the cartesian product of several variables. The potential is marginalized on space $X \cap Y$ and then vacuously extended on space Y. Plain marginalization is achieved when $Y \subseteq X$, and plain extension is achieved when $X \subseteq Y$. The operation corresponds to the uparrow in Shenoy's notation. It means we compute $m^{X \uparrow Y}$.

This function requires that the initial potential has at least one of $\{m, wc, wd\}$. You enter successively

1. the name of the initial belief defined on X,
2. the list of those variables included in Y. You get two list, at right, those in X, and at left, also those in X. You change this second list with the multi-select key (Right-Click or Cmd-Click) and construct the Y list.

Results are stored in a belief object which short (full) name is ChFr_xxx where xxx is the short (full) name of the belief defined on X.

6.5.9 Approximate Distinct Combination

When the number of focal elements is too large, approximation can be used to reduce computational load.

NA

6.5.10 Compare 2 belief potentials

When belief can be computed in two ways, we compare them to check the results are equal. Convenient to check algorithm correctness.

NA

6.5.11 EvClus

EvClus means ‘evidential clustering’. Algorithm of Denoeux-Masson.

Input is a dissimilarity matrix stored in folder User_Data/Other_Data. A demo file is catcortex.mat. You are asked to open the file with the data. The file is binary file .mat created by another MATLAB function. The file contains a matrix X(i,j), i= 1,...N, j = 1,...N, where X(i,j) is the dissimilarity between object i and object j.

It is assumed :

- individual can be categorized into N classes (an input parameter),
- for each individual, there is a belief potential (to be determined) representing our knowledge about the class to which it belongs,
- for every pair (i, j) of individuals, let $pl\{i, j\}(S)$ be the plausibility that they belong to the same class (computed from last belief potentials,
- we know the dissimilarity $d(i, j)$ between every pair of individuals (entered in the input matrix),
- the condition: $d(i, j) > d(k, l) \rightarrow pl\{i, j\}(S) < pl\{k, l\}(S)$.

The EvClus algorithm determines the belief potentials that satisfy (as well as possible) these conditions.

TBMLAB presents also a plot with the first two components of a multidimensional scaling plot.

Meaning of the parameters:

- option = 0, focal sets are singletons,
- option = 1, focal sets are singletons, the empty set and the universe,
- option ¿ 1, each belief potential can have up to 2^N focal sets.
- lambda is a regularity parameter, default = .001

Details, see: T. Denoeux and M. Masson. EVCLUS: Evidential Clustering of Proximity Data. IEEE Trans. SMC B, 2002.

6.5.12 Preferences

Default System initial preferences are installed.

Open Past preferences that have been saved are reloaded.

Save Save your preferences for future sessions.

GUI You can modify some of the parameters of the GUI's.

TBM You can modify some of the parameters of the TBMLAB output display.

Help Level There are 3 Help levels.

- max : you get all helps
- med : you don't get the button helps, only the window helps.
- min : no help is provided. Default option.

6.6 The Tutorials menu

The options available in the Tutorials menu are:

1. Mobius Transform
   (a) Potential Demo
   (b) Mobius Transf Operators
   (c) Explaining FMT
   (d) Specialization Demo
2. GBT Demo
3. Tutorials

6.6.1 Möbius Transforms

Explaining what are the Möbius Transforms.

Potential Demo

A tabulator for computing the various forms of potentials (m, b, Bel, pl, q, wc, wd, BetP). You enter one form and get the other forms of potentials.

Möbius Transform Operators

We show what are the matrices that transform one form of potential into another.

The matrices are built with a Kronecker product. $KRON(X,Y)$ is the Kronecker tensor product of $X = [x(i,j)], i = 1...I, j = 1...J$, and Y where X and Y are matrices. The result is a matrix equal to:

$$\begin{array}{lll} [x(1,1)Y & x(1,2)Y... & x(1,J)Y \\ x(2,1)Y & x(2,2)Y... & x(2,J)Y \\ ... & ...... & ... \\ x(I,1)Y & x(I,2)Y...x(I,J)Y]. & \end{array}$$

Each matrix is obtained with the next algorithm.

```
mat =1;
for i = 1:cardinal
        mat = Kron(mat0,mat);
end
```

where mat0 is one of the six matrices presented in table 6.6

Table 6.6: Basic generators of the Möbius transforms.

m_from_b		m_from_q		b_from_m	
1	0	1	-1	1	0
-1	1	0	1	1	1
m_from_q		**q_from_m**		**q_from_b**	
1	-1	1	1	0	1
1	0	0	1	-1	1

It is possible to write all the matrices x_frm_y with x,y∈ $\{m, b, bel, pl, q\}$. The only difficulty are with the bel or pl into m, b or q where the transformation contains an extra term. For the normalized forms, extra step for the transformations is elementary. For the *wc* and *wd* functions, we must pass through a log/exp transformation before using the matrices.

Explaining the Fast Möbius Transforms (FMT)

bel, pl, q are Möbius transforms of m, etc.... The FMT are to the inter potential form transformations what the FFT is to the Fourier transform We explain with an illustrative example what are the algorithms of the FMT.

The FMT is a very efficient algorithm to transform the various potential forms into each others.

For example, with cardinals 3 to 6, the numbers of additions required to transform m into b with the FMT and with the brute force method are given in table 6.7.

cardinal	FMT	brute force
3	12	12
4	32	50
5	80	180
6	192	602

Table 6.7: Numb XX er XX of additions to perform a Möbius transformation.

The only weakness of the FMT is that it requires to store a vector of length $2^{cardinal}$. So the algorithm is OK up to cardinal = 15, besides other algorithms less memory greedy might be more efficient. With sparse matrices (lot of zeros), one can hope for about $n^2/4$ additions where n is the number of focal sets (rule of thumb).

There is an algorithm for all the possible transforms where the coefficients are always 0, 1 and -1

Specialization Demo

A demo illustrating what are the specialization and Dempsterian specialization matrices.

The conjunctive impact of a piece of information can be represented by a specialization matrix. The conjunctive combination rule for distinct pieces of evidence is represented by a Dempsterian specialization matrix. We show how the algorithm works for a frame with 3 elements and for the m to b transformation.

6.6.2 GBT Demo

This demo explains what is the Generalized Bayesian Theorem (GBT).

Given the conditional belief $\beta^X[h_i], i = 1, ..., n$, over an observation space X for each element h_i of a frame $H = \{h_1, ..., h_n\}$ of hypothesis, find the belief induced on H given one has observed $x \subseteq X$. When the conditional belief can be accepted as induced by distinct pieces of evidence, the result is computed with the GBT

We illustrate the application of the GBT to a case where card$(X) = 3$ and card$(H) = 5$. Conditional bba can be entered as random vectors or from the keyboard.

6.6.3 Tutorials

In the Open File window, select the tutorial you want to run.

At right, you get a Tabulator window that displays the results of the computation performed during the tutorial.

At left, you get a Window with the tutorial text, and at bottom, red buttons with:

- quit : to quit the tutorial
- next : to display the next text
- Up, Down, Left, Right : to scroll the text

The menu bar contains:

- menu File/Quit. Performs the same action as the quit red button,
- menu Section. You get the list of the sections of this tutorial, and when selecting one, you jump directly to it.

6.7 The Help menu or TBM Help menu

NB: on Mac with MATLAB 5.2, this menu is called TBM Help, the other Help menu belongs to Mac and cannot be removed.

The options available in the Help menu are:

1. Menu Actions
2. Frame

3. Structure
4. Mapping
5. Attribute
6. Variable
7. Belief
8. About TBMLAB
9. Help Texts

Details of each option are given in the next subsections.

6.7.1 Menu Actions

A help explaining the tasks performed by each menu. You get a list of every menu. You select the one for which you want to get details.

6.7.2 Objects

Explaining what are the objects.

6.7.3 Frame

The help for the Frame object.

6.7.4 Structure

The help for the Structure object.

6.7.5 Mapping

The help for the Mapping object.

6.7.6 Attribute

The help for the Attribute object.

6.7.7 Variable

The help for the Variable object.

6.7.8 Belief

The help for the Belief object.

6.7.9 About TBMLAB

The welcome text is displayed on the Root Window.

6.7.10 Help Texts

All help texts are printed. Can be used as a note book.

6.8 The belief potential editor

You receive six panels, each with several options. Which buttons are displayed depend on the kind of variable on which the potential is build (discrete, continuous or integers). Only the displayed buttons are accessible. Each can be selected by clicking on its buttons. Four panels deal with the construction of the belief potentials, one with their display and one with the control option of the whole window.

1. Base on focal sets described by:
   - Sel.elem.: Cont/Int
   - Sel.elem.: No Cont/Int
   - Logical expression
   - Based on pdf: Cont/Int
2. Base on related functions:
   - BetP on elements
   - Pl on elements
   - With .m function
   - With random masses
3. Base on LCP and some constraints on:
   - bel
   - pl
   - wc
   - wd
4. Base on LCP and some Interval-value constraints on
   - m
   - bel
   - pl
   - wc
   - wd
5. View a potential
   - Largest m + focal
   - Bel,BetP,Pl,bba
   - Best supported set
   - Several forms

6. Control
    - Save data
    - Another belief
    - Get next potential
    - Quit

6.8.1 Base on Focal Sets

Selected elements: Continuous/Integers

You introduce the selected masses one by one, with their focal sets. This is the option to use when there are continuous and/or integers variables in the belief domain. In fact the only difference between a continuous and an integer variable resides in the way they are displayed (decimal or not).

TBMLAB displays the elements of the discrete part of the frame of discernment, one per line, in a scrolling window, and the limits of the intervals for the continuous/integer variables, one set of interval per element (see section 6.1.8. If there are no discrete variables, there is only one line with the set of limits for the other variables.

You click in multiple selection mode on those elements you want to put in the focal set under definition. You enter the limits for each cont/int variables in the query panel. When through, select the 'FocalSet: Done' button (bottom left).

Then you put a mass in the bottom box. Click on red button with 'mass OK'. You cycle. The total of the introduced masses is displayed. If you select no element in the list, the mass is given to the empty set.

When you click on the red button 'bba OK', you exit the construction of this potential. The mass to be given to the universe must not be introduced and TBMLAB computes automatically what has not been allocated, and the remaining mass is given to the universe.

The cancel button allows you to stop the construction of the bp. It get you back at the state before starting editing this potential (as if you had done nothing).

Selected elements: No Continuous/Integers

There are two modes to edit a bba with only discrete variables: set mode or logical expression mode. You select the one you want in the panel.

Set Mode. A $r \times c$ table: row = an element of the frame of discernment, column = one focal set. The left most column is the description of the elements of the frame of discernment. The top row = focal sets numbers and their masses. In the table, an x (.) means that the element of the row belongs (does not belong) to the focal set of the column. Horizontal and vertical lines are just given to help the user in keeping good parallelism.

By clicking (right-click on PC) on the x or ., their values toggle, hence you can enter an element into a focal set or delete an element from a focal set. By clicking on the mass, you get a window where you enter the next new mass, then hit RETURN key.

The control buttons at the bottom:

- up, down, left and right : for scrolling the tables.
- Cancel : you cancel the whole editing process, you get back the previous bba.
- Quit : you end the editing process, and all changes are stored in the belief object (but not on disc)
- Add : a new focal set is created with mass 0 and set = empty set.

To delete a focal set, change its mass into 0, it will be deleted when you apply Quit.

Logical Expression Mode. If there are no logical expressions, you are send directly into the set mode.

List: for each focal set, its number, its mass, its logical expression when there is one. You select the focal set to be edited (only those with a logical expression). For each selected case, you get the Logical Expression Editor window (the one used for creating a logical expression). You can clear the expression and create a new one, type into the formula (dangerous), put the cursor in the formula and use the menu buttons (dangerous too). You can also just keep it and change the mass. Change are saved once you select Quit and Save

To edit only the masses, use the Set Mode, it is simpler.

Logical Expression

User can describe the focal sets with logical expressions, and give a mass to each of them. It is not restricted to propositions, but works for any variable.

A typical logical expression can be:

$$\text{var1} \in \{x, y, z\} \wedge (\text{var2} = 3 \vee \neg p) \rightarrow \text{var3} \in \{a, b, c, d\}$$

On your screen, it will show up as;

(var1 in {'x','y','z'}) and (var2 in [3] or neg p) → (var3 in {'a','b','c','d'})

You get

1. a list of all variables involved in the present belief object. TBMLAB adds two extra variable called universe and contradiction.
2. a list of logical operators: and, or, not, implies, equivalent, (,).
3. a box at bottom where the logical expression will be displayed.

You select the variables and the logical operators in order to build a well defined formula that will appear in the bottom box. If the variable is not of logical type, you get a vertical list of its elements (the elements of its frame), and you multi-select those you needed for a proposition $var \in \{\dots\}$. You finish the expression by selecting the OK button. You enter the related mass, then click again on the OK button. You repeat for every mass.

A logical expression can be erased with Clear Expression.

To finish entering one belief potential with the logical expression option, click on ‘Quit and Save’ if you agree with what you did.

The unallocated mass is given to the ‘tautology’.

The button Cancel No Save lets you quit the logical expression editor and nothing done since entering this window is saved.

Don’t try to type in the box. You are NOT supposed to do it. It is not forbidden, but you do it at your own risk. Not knowing TBMLAB syntax will make it hard for beginners. The real trick is ‘put space on both side of every operator’.

The rules for building well defined formulas are those of propositional logic. The atomic terms are:

- a logical variable, thus created as a proposition (like when using Logical propositions in the Edit/Short Cuts menu)
- (var_name in {’val1’,’val2}) which means that the symbolic value of variable var_name must be one of ’val1’, ’val2’ ...
- (var_name in [14 7 1:4 23]) which means that the numerical value of variable var_name must be 14 or 7 or 1 or 2 or 3 or 4 or 23.

The operators are the logical operators not, and, or, impl (for implies) equiv (for equivalent). Parenthesis are used as in logic to fix order precedence. Default precedence for and, or and not is as in logic. ‘p impl q’ is transformed into ‘not p or q’ and ‘p equiv q’ is transformed into‘ p and q or (not p and not q)’. For impl and equiv, you may not use more than one impl or one equiv in the whole formula.

The operators apply to formulas. Formulas are a term, operator separated terms, operator separated formulas.

Example. Suppose a frame made as the product of 4 variables. Let them be two logical variables p and b and V_ABC : a variable with 3 possible values A B and C,, and N_100 : a variable which values are the integers from 0 to 100.

Example of valid formulas:

- p
- p or b
- p impl not b
- p equiv b
- not (p impl b)
- not (V_ABC in {’A’,’C’}) impl p
- p and not b and (N_100 in [40:50 60:80 99])

Note that the options like 60:70 need some extra typing in the box area. You select (at least) one element for the variable, so TBMLAB prepare the (var in [...]) structure. You can erase what is between [and] and replace it by lists like 60:70 etc... What is between [and] is a list handled as such by MATLAB.

Note: This method is efficient to enter product sets. Suppose there are two integer variables X and Y, each with values from 0 to 100. To select the set $(x \times y) = \{(x_i, y_j) : x_i \in x, y_j \in y\}$ where $x = \{5, 6, 7\}, y = \{12, 13, 17\}$ you create the logical expression (X in [5,6,7]) and (Y in [12 13 17]).

Based on pdf: Cont/Int

Bba created by a .m function, or derived from a pdf. For Continuous/Integers only. XXXX

6.8.2 Base on related functions

BetP on elements

For those elements where BetP is positive, you introduce the value of BetP and its related singleton. TBMLAB then computes the least committed belief potential (an application of the LCP, least committed principle) that would produce these pignistic probabilities. TBMLAB stores the masses and their focal sets. The potential so created is called the isopignistic least committed belief potential generated by BetP.

In practice, TBMLAB lists all the elements of the frame of discernment of the present potential. At bottom of the window, you get a box and two buttons. You put a BetP value is the box, and select one element in the list of elements. Click on red button with mass OK. The total allocated probability is updated and displayed at bottom of UR. Cycle up every probabilities are allocated. Quit by clicking on red button, BetP OK.

Unallocated probability (when sum of allocated BetP < 1). The remainder is equally distributed among all the elements of the frame of discernment. It is a convenient way to generate a potential with equi probability on every element, even though it is still easier to use the selected m option and just return when asked for the first mass.

Mathematic. Let $\Omega = \{\omega_1, \dots \omega_n\}$ be a frame of discernment with $BetP(\omega_i)$ the pignistic probabilities allocated to its elements, and the elements of Ω be so ordered that $BetP(\omega_i) \geq BetP(\omega_{i-1})$ for $i = 1, \dots |\Omega|$, defining $\omega_0 = \emptyset$ and $BetP(\omega_0) = 0$ (equality can be arbitrarily resolved, results will not be affected).

Define $A_1 = \Omega$.

Then computed iteratively for $i = 1, \dots, |A_i|$:

$$\begin{aligned} m(A_i) &= |A_i|(BetP(\omega_i) - BetP(\omega_{i-1})) \\ A_{i+1} &= A_i \cap \overline{\omega_i} \end{aligned}$$

The result are the basic belief masses of a consonant belief potential which related plausibility function is a possibility function.

Pl on Elements

TBMLAB selects each element by moving down the list from element to element. You enter the plausibility for each element. Default is 1, so when 1, just click OK. TBMLAB determines the least committed belief potential which plausibilities on the elements are those entered.

With .m function

User furnishes a .m function that MATLAB can execute. It will be evaluated by TBMLAB using eval(...). NA

With Random Masses

TBMLAB generates random masses on the subset of the frame of discernment (at most 500 subsets). Practical for learning purpose.

6.8.3 Base on LCP and some Constraints

User provides values of bel, pl, wc, wd on some subsets of the frame of discernment and TBMLAB builds the least committed belief potential compatible with these constrains.

6.8.4 Base on LCP and some Interval-value Constraints

As in the previous case, the user provides interval for the values of m, bel, pl, wc, wd on some subsets of the frame of discernment and TBMLAB builds the least committed belief potential compatible with these constrains.

6.8.5 View a potential

Not yet developed for Continuous and Integers variables. Wait next version.

Largest m + Focal

A plain list of each mass with its focal sets described by the list of elements belonging to the set. Masses are ordered. Either the largest are displayed first, or they are displayed in the order you have entered them.

We limit the number of masses so that 95% (also modifiable in Tools/Preference menu) of the total normalized masses are listed. Small ones are not very useful to list, and list can be already quite long.

At UR, you get the normalized mass given to the next set, the unnormalized mass given to the next set, the percent of mass already displayed, and the mass given to the empty set/contradiction. Then the focal set is described by the list of its elements and its generating logical expression if it exists. Separation between two masses is made of a line of -.

A more sophisticated presentation of these masses is given by the next option.

Bel, BetP, Pl, bba

You get:

- a plot with the values of Bel, BetP, Pl for each elements, ordered according to the BetP values, or the Pl values, or the elements numerical values in their binary representation. The number of elements displayed stops when 95% of the total probability has been displayed, when the gap between successive BetP is large (more than 40% by default) or when the number of elements displayed is above a certain limit (16 by default), whichever applies first. These parameters can be modified in the menus and also in the Tools/Preference menu.
- above the plot, a list with the 'meaning' of the indexes used in the plot to label the plotted elements.

- a list of the normalized masses where focal sets are presented as a list of sets of elements using the dot convention. An element is characterized by the value of its underlying variables. A dot means that every element built by assigning to the dotted variable all the possible values of the variable belongs to the focal set. So if $X = \{x_1, x_2\}$, $Y = \{y_1, y_2, y_3\}$, $Z = \{z_1, z_2\}$ and the elements are ordered as X, Y, Z, then

  $$(x_1, ., z_2) = \{(x_1, y, z_2) : y \in Y\}$$

  $$(., ., z_2) = \{(x, y, z_2) : x \in X, y \in Y\}.$$

  This notation often simplifies the set representation by shortening the list.

  The logical formula used to construct the focal set is given if the belief potential was built by using the logical expression option.

Best Supported Sets

Determination of the subset that should contain the actual value (reference = A. Appriou). For every subset A of the universe, compute $Pl(A)/|A|^r$ where Pl is the normalized plausibility and r is a parameter in [0,1] where $r = 0$ means select the universe, $r = 1$ means select the most plausible singleton, The larger r, the more singletons are favored. For each r, find out the best supported subset according to Appriou criteria. You can conclude that data support essentially that the actual value of the unknown parameter is in this set. If stable for all r, selection is clear, otherwise there is some left over ambiguity. Once a selection is made, you may conclude without much risk of error: the data support that the actual value is in the selected subset.

In order to avoid too long computation, it does not work if there are more than 50 elements.

Several forms

Full screen display of several requested forms for one potential.

It works only if cardinality of the domain_variable is ≤ 12. Think about the number of subsets when the cardinality is above 12 (larger than $2^{12} = 4096$. Why would you be interested in listing all the values of some function for so many values?

6.8.6 Control

Save data

All your objects are saved as per now.

Another belief

Display the list of belief objects. You can select another one.

Get next potential

Start using the next potential (useful only for conditional beliefs).

Quit

Return to the calling window.

6.9 The Representation of the Sets

Only for the discrete variables, not for the continuous and integer variables.

We explain how the focal sets are represented in the field belief(i).bp(j).form(k).focal_sets.

When building a belief object, you are asked to enter the list of variables on which the belief is built. This is done by a multiselection in the displayed list of variables. Suppose you select three of them, which numbers are 25, 33 and 71. TBMLAB stores these numbers as a list and the order is the order under which these variables were presented in the list (from top to bottom). Suppose that in the list the variable number 71 was listed before the variable number 25, itself before variable number 33. The vector [71 25 33] is then stored in belief(i).domain_variable = [71 25 33]. The cardinalities of these variables are stored in belief(i).domain_cardinality = [2 4 3]. Thus the cardinality of variable 71 is 2, for variable 25 it is 4, for variable 33 it is 3.

Given these data, we can thus build $2 \times 4 \times 3 = 24$ different worlds. Table 6.9 presents the list of these 24 worlds and the meaning of each of them. Thus world 13 is a world where variable 71 is 1, variable 25 is 3 and variable 33 is 2.

To build a set, select those worlds that belong to it. Suppose the set A with worlds 2 and 5. We build a list of 24 bits. The rightmost bit corresponds to world 1, the leftmost to world 24. Put 0 if the corresponding world does not belong to the set and 1 if it does. So the list for set A is presented in table 6.9.

This list of bits is considered as the binary representation of a number, which value is 18 for set A, and this number is stored in the focal_set field

Depending on the platform, an integer can store a limited numbers of bits we call number_bit_per_word (usually number_bit_per_word = 52). We decide to store up to (number_bit_per_word - 2) bits per word, thus usually 50. In that case and if there are 120 worlds, we store bits 1 to 50 in word 1, bits 51 to 100 in word 2, and the bits 101 to 120 in word 3. For the set B with worlds 2,5 and 101, the number stored in belief(i).bp(j).form(k).focal_sets(n,1:3) is [18 0 1].

Ordered Set Representation Whenever elements of a set are just ordered, we use the same binary order as above. So if there are three worlds denoted {a,b,c}, we have 8 worlds. The 8 subsets are ordered as follows: {}, {a}, {b}, {a,b}, {c}, {a,c}, {b,c}, {a,b,c}. Suppose we have masses given to these subsets, they are stored in a vector with 8 elements. The first element of the vector corresponds to the mass given to the empty set, the second to the mass given to the set {a}, the sixth to the mass given to the set {a,c}...

6.10 Appendix: the natural frames

List of the frames created in the Get natural frames option from Short Cuts.

- Logical

variable			
33	25	71	world number
3	4	2	
1	1	1	1
1	1	2	2
1	2	1	3
1	2	2	4
1	3	1	5
1	3	2	6
1	4	1	7
1	4	2	8
2	1	1	9
2	1	2	10
2	2	1	11
2	2	2	12
2	3	1	13
2	3	2	14
2	4	1	15
2	4	2	16
3	1	1	17
3	1	2	18
3	2	1	19
3	2	2	20
3	3	1	21
3	3	2	22
3	4	1	23
3	4	2	24

Table 6.8: Relation between the variables and the worlds.

Number :	1
short_name :	FT
full_name :	Logical_FT
Type :	Logical
Elements :	T F
Cardinality	2
Min	0
Max	0

- Binary

Number :	2
short_name :	B01
full_name :	Binary_01
Type :	Binary
Elements :	0 1

world	24	23	22	21	20	19	18	17	16	15	14	13
set A	0	0	0	0	0	0	0	0	0	0	0	0
world	12	11	10	9	8	7	6	5	4	3	2	1
set A	0	0	0	0	0	0	0	1	0	0	1	0

Table 6.9: Set A that contains worlds 2 and 5.

Number : 3
short_name : NY
full_name : No_Yes
Type : Binary
Elements : N
Y

Number : 4
short_name : MF
full_name : Male_Female
Type : Binary
Elements : M
F

Number : 5
short_name : FrFo
full_name : Friends_Foe
Type : Binary
Elements : Fr
Fo

Number : 6
short_name : PN
full_name : Pos Neg
Type : Binary
Elements : P
N

Number : 7
short_name : FA
full_name : For_Against
Type : Binary
Elements : F
A

Number : 8
short_name : AB
full_name : 2ary AB
Type : Binary
Elements : A
B

For each frame:

Cardinality	2
Min	0
Max	0

- Classes ordered or unordered.

Number :	11
short_name :	ABC
full_name :	UnOrd_Classes ABC
Type :	Classes_Un_ordered
Elements :	A
	B
	C
Cardinality	3
Min	0
Max	0

Number :	12
short_name :	F123
full_name :	Ord_Classes 123
Type :	Classes_Ordered
Elements :	C1
	C2
	C3
Cardinality	3
Min	0
Max	0

Number :	13
short_name :	ABCD
full_name :	UnOrd_Classes ABCD
Type :	Classes_Un_ordered
Elements :	A
	B
	C
	D
Cardinality	4
Min	0
Max	0

Number :	14
short_name :	F1234
full_name :	Ord_Classes 1234
Type :	Classes_Ordered
Elements :	C1
	C2
	C3
	C4
Cardinality	4

Min	0
Max	0

Chapter 7

The TBM

This chapter is an introduction to the TBM. It is useful to fix the definitions of the terms used in the TBMLAB user guide.

The literature about belief functions starts with the work of Dempster (Dempster, 1972, 1969, 1968a, 1968b, 1967a, 1967b, 1966).

Based on Dempster' work, Shafer invent the concept of belief functions as such, and develop the concepts of Dempster's rule of conditioning, disjunctive combination rule, discounting and commonality functions (Shafer, 1976).

See http://www.glennshafer.com/.

Kohlas developed the Hint model and the related PAS, which are based essentially on Dempster's ideas (Kohlas & Monney, 1995).

See http://diuf.unifr.ch/tcs/publications/personal/Kohlas.htm

In these works, the concept of probability was still present. Their models are some kind of extensions of probability theory., Smets considers that beliefs precede the concept of probability, that belief states are represented by belief functions, and that probabilities show up only once decisions are involved. The TBM (for transferable belief model) was really introduced in (Smets & Kennes, 1994) and a recent presentation an be found in (Smets, 1998). Many new concepts were developed within the TBM.

See http://iridia0.ulb.ac.be/ psmets/

For the algorithmic part, the major work was done by Shenoy who, in collaboration with Shafer, invents the concepts of the valuation based systems (VBS) which is an efficient algorithm for the propagation of beliefs in networks.

See http://lark.cc.ukans.edu/ pshenoy/

For applications, see http://www.hds.utc.fr/ tdenoeux/

7.1 Introduction

7.1.1 The representation of quantified beliefs

The TBM is a mathematical model to represent the beliefs held by an agent about the value of the actual world.

The frame of discernment. We start with a set of worlds $\Omega = \{\omega_1; \omega_2, \dots\}$ called the frame of discernment. Usually Ω is finite. By default Ω will be finite.

The actual world. One of the worlds in the frame of discernment, denoted ω_0, corresponds to the actual world.

The term ‘world’ is used in a general sense. It covers concepts like ‘alternative’, ‘hypothesis’, ‘state of affairs’, ’state of nature’, ‘situation’, ‘context’, ‘value of a variable’ ...

Note. Under the ‘closed world assumption’, $\omega_0 \in \Omega$. Under the ‘open world assumption’, ω_0 might not be in Ω. In any case ω_0 is unique.

The agent You. There is an agent, denoted You (but it might be a robot, a piece of software, a computer program ...) who is the belief holder. You do not know which world in Ω corresponds to the actual world ω_0. Nevertheless, You have some idea, some opinion about which world might be the actual one. So for every subset A of Ω, You can express the strength of Your opinion that the actual world ω_0 belongs to A. This strength of opinion is called the degree of belief, or the belief for short. The larger the belief in A, the stronger You believe $\omega_0 \in A$.

The time. Beliefs are held by You at a given time t and can of course change with time, hence the t index.

The static and dynamic components. Any model that wants to represent quantified beliefs has, at least, two components:

- the static component that describes Your state of belief at time t given the information available to You,
- the dynamic component that explains how to revise Your beliefs given new pieces of information become available to You at t.

Credal versus pignistic levels. We have described a two level mental model in order to distinguish between two aspects of beliefs, belief as weighted opinions, and belief for decision making. The two levels are:

- the credal level, where beliefs are entertained, and
- the pignistic level, where beliefs are used to make decisions.

The terms credal and pignistic derive from the latin words ‘credo’, I believe and ‘pignus’, a wage, a bet.

Beliefs at the credal level are represented by belief functions. When decision must be made, the belief held at the credal level induced (through the pignistic transformation) a pignistic probability at the pignistic level, and decisions are taken by using the expected utility computed with the pignistic probability function.

The evidential corpus. The evidential corpus $EC_{You,t}$ is a set of propositions accepted as true by You at t. In practice, we put in $EC_{You,t}$ only those propositions relevant to Your beliefs at t about the variable for which beliefs are considered.

7.2 The static component

For a given variable, the evidential corpus $EC_{You,t}$ held by You at t induces a belief state about the actual value of the variable. This belief state about the value of the actual world $\omega_0 \in \Omega$ is denoted $BS^{\Omega}_{You,t}[EC_{You,t}]$. The set of belief states on Ω is denoted $\mathcal{B}^{\Omega}$.

A belief potential ϕ^{Ω} relative to Ω is a mapping from $\mathcal{B}^{\Omega}$ and a set of symbols $\mathcal{S} = \{m, b, bel, pl, q, wc, wd, M, Bel, Pl, Q\}$ to a vector in $R^{2^{|\Omega|}}$ with for instance $\phi^{\Omega}(BS^{\Omega}_{You,t}[EC_{You,t}], m)$ being the bba $m^{\Omega}_{You,t}[EC_{You,t}]$. In general $\phi^{\Omega}(BS^{\Omega}_{You,t}[EC_{You,t}], x) = x^{\Omega}_{You,t}[EC_{You,t}]$ where $x \in \mathcal{S}$.

The degree of belief held by agent You at time t about the fact that the actual word ω_0 is an element of $A \subseteq \Omega$ is denoted $bel^{\Omega}_{Yoy,t}(\omega_0 \in A)$ or $bel^{\Omega}_{Yoy,t}(A)$. When obvious from the context, You and t labels are neglected. Even the frame of discernment Ω can also be neglected, what leads to a notation like $bel(A)$.

The TBM departs from the Bayesian approach in that we do not assume the additivity required in probability theory. It is replaced by inequalities like:

$$bel(A \cup B) \geq bel(A) + bel(B) - bel(A \cap B). \tag{7.1}$$

A belief function bel is a function from 2^{Ω} (the power set of the frame of discernment Ω) to [0,1] that satisfies:

$$bel(\emptyset) = 0 \tag{7.2}$$

$$\forall n > 1, \forall A_1, A_2, \ldots A_n \subseteq \Omega, bel(A_1 \cup A2 \cup \ldots A_n) \geq$$

$$\sum_i bel(A_i) - \sum_{i>j} bel(A_i \cap A_j) \cdots - (-1)^n bel(A_1 \cap A_2 \cap \ldots A_n). \tag{7.3}$$

As such, the meaning of these inequalities is not obvious, except when $n = 2$ (see (7.1)).

7.2.1 Basic belief assignment

The understanding of the inequalities (7.3) is clarified once the concept of a basic belief assignment is introduced.

Basic belief assignment. A basic belief assignment (bba) is a function $m : 2^{\Omega} \to [0, 1]$ that satisfies

$$\sum_{A:A\subseteq\Omega} m(A) = 1. \tag{7.4}$$

Basic belief mass. The term $m(A)$ is called the basic belief mass (bbm) given to A.

Focal element. Any $A \subseteq \Omega$ such that $m(A) > 0)$ is called a focal element of m.

Definition of $m(A)$. The bbm $m(A)$ represents that part of Your belief that supports A — i.e., the fact that the actual world ω_0 belongs to A — without supporting any more specific subset, by lack of adequate information.

7.2.2 Belief functions

The bbm $m(A)$ does not in itself quantify Your belief, denoted by $bel(A)$, that the actual world ω_0 belongs to A. Indeed, the bbm $m(B)$ given to some subset B of A also supports that $\omega_0 \in A$ without supporting $\omega \in \overline{A}$. Hence, the degree of belief $bel(A)$ is obtained by summing all the bbm's $m(B)$ for $B \subseteq A, B \neq \emptyset$. We have:

$$\begin{aligned} bel(A) &= \sum_{B:\emptyset \neq B \subseteq A} m(B) \qquad \forall A \subseteq \Omega, A \neq \emptyset \\ bel(\emptyset) &= 0. \end{aligned} \tag{7.5}$$

Definition of $bel(A)$. The degree of belief $bel(A)$ for $A \subseteq \Omega$ quantifies the total amount of *justified specific support* given to the fact that the actual world ω_0 belongs to A.

Note: Shafer assumes $m(\emptyset) = 0$, or equivalently $bel(\Omega) = 1$. In the TBM, such a requirement is not assumed.

7.2.3 Plausibility functions

The dual of bel is called a plausibility function $pl : 2^\Omega \rightarrow [0, 1]$. It is defined as:

$$pl(A) = bel(\Omega) - bel(\overline{A}), \quad \text{for all } A \subseteq \Omega,$$

or

$$pl(A) = \sum_{X \subseteq \Omega : X \cap A \neq \emptyset} m(X), \quad \text{for all } A \subseteq \Omega.$$

Definition of $pl(A)$. The degree of plausibility $pl(A)$ for $A \subseteq \Omega$ quantifies the maximum amount of potential specific support that could be given to the fact that the actual world ω_0 belongs to A.

7.2.4 Special belief functions

Vacuous belief function. The vacuous belief function satisfies $m^\Omega(\Omega) = 1$. It represents a state of total ignorance.

Categorical belief function. A categorical belief function is a belief function with one focal element $X \neq \Omega$, thus $\exists X \subseteq \Omega, X \neq \Omega$ and $m^\Omega(X) = 1$ It represents a state of full but imprecise (except when $|X| = 1$) knowledge.

A simple support function. A simple support function is a belief function with two focal sets, one being Ω. Thus $\exists X \subseteq \Omega, X \neq \Omega$ and $m^\Omega(X) = 1 - x, m^\Omega(\Omega) = x, x \in [0, 1]$. It is also denoted as X^x.

The contradictory belief function. The contradictory belief function is a categorical belief function where $m^\Omega(\emptyset) = 1$. It represents full contradiction.

Consonant Belief functions. A consonant belief function is a belief function which focal sets are nested. Thus there is an indexing of the focal sets such that:

$$X_1 \subseteq X_2 \subseteq X_3 \ldots \subseteq X_n$$

where the X_i's are the focal sets of m. In that case,

$$b(A \cap B) = \min(b(A), b(B)) \tag{7.6}$$

$$bel(A \cap B) = \min(bel(A), bel(B)) \tag{7.7}$$

$$pl(A \cup B) = \max(pl(A), pl(B)) \tag{7.8}$$

The function *bel* is a necessity function and the function *pl* is a possibility function.

Non-dogmatic belief functions. A non-dogmatic belief function on Ω is a belief function with $m^\Omega(\Omega) > 0$.

Normalized belief functions. A normalized belief function on Ω is a belief function with $m^\Omega(\emptyset) = 0$. Normalized potentials are denoted as M, Bel, Pl, Q.

7.2.5 Other related functions

Other functions related to *bel* are also defined.

The commonality function. $q : 2^\Omega \to [0,1]$ with:

$$q(A) = \sum_{X \subseteq \Omega : A \subseteq X} m(X), \quad \text{for all } A \subseteq \Omega.$$

The implicability function. $b : 2^\Omega \to [0,1]$ with:

$$b(A) = bel(A) + m(\emptyset) = \sum_{X \subseteq \Omega : X \subseteq A} m(X), \quad \text{for all } A \subseteq \Omega.$$

The conjunctive canonical weights. $wc : 2^\Omega \to [0,\infty]$ with:

$$wc(A) = \prod_{B \subseteq \overline{A}} q(A \cup B)^{-1^{1+|B|}} \quad \text{for all } A \subseteq \Omega.$$

It is well defined for non-dogmatic belief functions.

The disjunctive canonical weights. $wc : 2^\Omega \to [0,\infty]$ with:

$$wd(A) = \prod_{B \subseteq A} b(B)^{-1^{1+|A]-|B|}} \quad \text{for all } A \subseteq \Omega.$$

It is well defined for belief functions with some contradiction ($m(\emptyset) > 0$).

Their major interest will appear when conditioning and combination will be introduced.

It must be emphasized that each of these functions $(m, b, bel, pl, q, wc, wd)$ are in one-to-one correspondence, so none gives an information not included in the others. Their interest comes from the fact they enhance different aspects of the same underlying belief state, and that some are sometime more convenient.

The Möbius transforms. The transformations between m, b and q are called Möbius transformations.

7.3 Dynamic Components

7.3.1 Discounting

Suppose a second agent Me. You communicate Me Your belief about Ω, represented by $m^{\Omega}_{You,t}$. If I consider You are fully reliable, I would accept Your bba as mine, thus $m^{\Omega}_{Me,t} = m^{\Omega}_{You,t}$. But I am not ready to accept it and I consider that You are only partially reliable. Let $\alpha \in [0,1]$ be My belief that You are reliable.

Then My belief $m^{\Omega}_{Me,t}$ is $Yours$ discounted by a factor α

$$\begin{aligned} m^{\Omega}_{Me,t}(X) &= \alpha m^{\Omega}_{You,t}(X), \ \forall X \neq \Omega \qquad (7.9) \\ &= \alpha m^{\Omega}_{You,t}(\Omega) + 1 - \alpha. \qquad (7.10) \end{aligned}$$

We say $m^{\Omega}_{Me,t}$ is $m^{\Omega}_{You,t}$ discounted by the factor $(1-\alpha)$ and $(1-\alpha)$ is called the discount rate.

7.3.2 Specialization

Many rules for revising beliefs are based on the concept of specialization.

Let m_0 be the bba on Ω held by You at time t_0. The value $m_0(A)$ is that part of Your belief that supports $A \subseteq \Omega$ and does not support any strict subset of A due to lack of information. If further information obtained by You at time t_1 with $t_1 > t_0$ justifies it, the basic belief mass $m_0(A)$ that was supporting A at t_0 might support more specific subsets of A. This fits in with the idea that $m(A)$ was not allocated to subsets more specific than A by lack of information. When new information is obtained, $m(A)$ might thus 'flow' to subsets of A, and it may not move outside of A as we already knew that it specifically supports A. Therefore, the impact of a new piece of evidence results in a redistribution of $m_0(A)$ among the subsets of A. This redistribution can be characterized by a set of non negative coefficients $s(B,A) \in [0,1], A, B \subseteq \Omega$, where $s(B,A)$ is the proportion of $m_0(A)$ that is transferred to B once the new piece of evidence is taken into account by You. The s coefficients depend of course on the piece of evidence that initiated the belief revision.

In order to conserve the whole mass $m_0(A)$ after this transfer, the $s(B,A)$ must satisfy:

$$\sum_{B \subseteq A, B \subseteq \Omega} s(B,A) = 1 \qquad \forall A \subseteq \Omega \qquad (7.11)$$

As masses can only flow to subsets, $s(B,A) = 0$ for all B not included in A. The matrix $\mathbf{S}$ of such coefficients $s(B,A)$ for $A, B \subseteq \Omega$ is called a specialization matrix on Ω.

After You learn the new piece of evidence E, Your initial basic belief assignment m_0 is transformed into the new basic belief assignment m_1 such that:

$$m_1(A) = \sum_{X \subseteq \Omega} s(A,X) m_0(X) \qquad (7.12)$$

This formula reflects the idea that the bbm $m_0(X)$ initially allocated to X is distributed among the subsets of X after applying the specialization operator. This down-flow reflects the meaning of $m_0(X)$ as the part of belief that specifically supports X, but might support more specific subsets if further information justifies it.

The basic belief assignment m_1 is called a specialization of m_0. For a bba m, we use the notation $\mathbf{m}$ to represent to column vector with elements $m(A)$ for $A \subseteq \Omega$. Relation (11) can be written as:

$$\mathbf{m}_1 = \mathbf{S} \cdot \mathbf{m}_0 \tag{7.13}$$

If the basic belief assignment m_1 is a specialization of the basic belief assignment m_0, then m_1 is at least as committed as m_0.

Example 7.3.1.[An example of specialization] Let $\Omega = \{a, b, c\}$. Suppose a bba $\mathbf{m}_0$ defined on 2^Ω. Let $\mathbf{S}$ be a specialization matrix, and let $\mathbf{m}_1 = \mathbf{S}.\mathbf{m}_0$. Table 1 presents the values of $\mathbf{S}$, $\mathbf{m}_0$ and $\mathbf{m}_1$. We use the iterated order illustrated in table 1 to list the elements of the vectors $\mathbf{m}$ and the specialization matrix $\mathbf{S}$.

Table 7.1: Values of the specialization matrix $\mathbf{S}$, and of the bba $\mathbf{m}_0$ and $\mathbf{m}_1$ with $\mathbf{m}_1 = \mathbf{S}.\mathbf{m}_0$. The blanks in the $\mathbf{S}$ matrix indicate those values of $\mathbf{S}$ that must be null so that $\mathbf{S}$ is a specialization matrix.

$$
\begin{array}{r|c|c|cccccccc|c|c|}
\multicolumn{1}{r}{} & \multicolumn{1}{c}{\mathbf{m}_1} & \multicolumn{1}{c}{=} & & & & & \mathbf{S} & & & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{\mathbf{m}_0} \\
\multicolumn{1}{r}{} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} & \varnothing & \{a\} & \{b\} & \{a,b\} & \{c\} & \{a,c\} & \{b,c\} & \multicolumn{1}{c}{\{a,b,c\}} & \multicolumn{1}{c}{} & \multicolumn{1}{c}{} \\
\varnothing & .13 & & 1. & .3 & .5 & .2 & .0 & .4 & .0 & .2 & & .0 \\
\{a\} & .04 & & & .7 & & .5 & & .1 & & .1 & & .0 \\
\{b\} & .13 & & & & .5 & .0 & & & .4 & .0 & & .1 \\
\{a,b\} & .00 & = & & & & .3 & & & & .0 & \cdot & .0 \\
\{c\} & .50 & & & & & & 1. & .3 & .0 & .5 & & .3 \\
\{a,c\} & .00 & & & & & & & .2 & & .0 & & .0 \\
\{b,c\} & .12 & & & & & & & & .6 & .0 & & .2 \\
\{a,b,c\} & .08 & & & & & & & & & .2 & & .4
\end{array}
$$

□

7.3.3 Conditioning

Suppose You have some belief on Ω represented by the basic belief assignment m^Ω. Then a new piece of evidence becomes available to You and it implies that the actual world cannot be in one of the worlds in $\overline{A}$. Then the mass $m^\Omega(B)$ that initially was supporting that the actual world is in B now supports that the actual world is in $B \cap A$ as every world in $\overline{A}$ must be 'eliminated'. So $m^\Omega(B)$ is transferred to $B \cap A$ after conditioning on A. (The model gets its name from this transfer operation.) This rule is called **Dempster's rule of conditioning**.

$$
\begin{aligned}
m^\Omega[A](B) &= \sum_{C:C\subseteq\overline{A}} m^\Omega(B\cup C) && && (7.14)\\
b^\Omega[B](A) &= b^\Omega(B\cup\overline{A}) && && (7.15)\\
bel^\Omega[B](A) &= bel^\Omega(B\cup\overline{A}) - bel^\Omega(\overline{A}) && && (7.16)\\
pl^\Omega[B](A) &= pl^\Omega(B\cap A) && && (7.17)\\
q^\Omega[A](B) &= q^\Omega(B) && \text{if } B\subseteq A, && (7.18)\\
&= 0 && \text{otherwise} && (7.19)
\end{aligned}
$$

7.4 Non interactive combination rules

Notation: remember the symbols between [and] denote the pieces of evidence taken in consideration when building Your belief function on a frame Ω. So $bel[E_1 \wedge E_2]$ is the belief function built when both the pieces of evidence E_1 and E_2 are taken in consideration. $bel[E_1](A)$ denotes then the value of the belief function $bel[E_1]$ at $A \subseteq \Omega$.

7.4.1 Belief functions induced by distinct pieces of evidence

All pieces of evidence are considered as 'distinct'. Mathematically, it means that bel^Ω induced by E_1 and E_2 is a function of $bel^\Omega[E_1]$ and $bel^\Omega[E_2]$, the belief functions induced by E_1 and E_2, respectively.

The conjunctive combination rule

Suppose two pieces of evidence E_1 and E_2, and You consider that both hold and their sources are fully reliable.

Let $m_1^\Omega = m^\Omega[E_1]$ and $m_2^\Omega = m^\Omega[E_2]$ denote the bba induced by E_1 and E_2, respectively.

The belief function that results from the conjunctive combination of m_1^Ω and m_2^Ω can be denoted equivalently

- $m^\Omega[E1 \wedge E2]$ or $m^\Omega[E1, E2]$
- $m_1^\Omega \textcircled{$\cap$} m_2^\Omega$ or $m_{1\textcircled{$\cap$}2}^\Omega$

where the $\wedge$ is the logical 'and', and the $\textcircled{$\cap$}$ operator denotes the conjunctive combination.

We have

$$
\begin{aligned}
m_{1\textcircled{$\cap$}2}^\Omega(C) &= \sum_{A\subseteq\Omega, B\subseteq\Omega, A\cap B=C} m_1^\Omega(A)m_2^\Omega(B), && \forall C\subseteq\Omega && (7.20)\\
q_{1\textcircled{$\cap$}2}^\Omega(A) &= q_1^\Omega(A)q_2^\Omega(A) && \forall A\subseteq\Omega && (7.21)
\end{aligned}
$$

The Dempster's rule of combination

Dempster's rule of combination is the same as the conjunctive combination rule except the end result is normalized. The $\text{\textcircled{$\cap$}}$ symbol is replaced by $\oplus$.

$$m^{\Omega}_{1\oplus 2}(C) = \frac{m^{\Omega}_{1\text{\textcircled{$\cap$}}2}(C)}{1 - m^{\Omega}_{1\text{\textcircled{$\cap$}}2}(\emptyset)} \tag{7.22}$$

$$q^{\Omega}_{1\oplus 2}(A) = \frac{q^{\Omega}_{1}(A) q^{\Omega}_{2}(A)}{1 - m^{\Omega}_{1\text{\textcircled{$\cap$}}2}(\emptyset)} \forall A \subseteq \Omega \tag{7.23}$$

This definition holds only if $m^{\Omega}_{1\text{\textcircled{$\cap$}}2}(\emptyset) < 1$.

The disjunctive combination rule

Suppose two pieces of evidence E_1 and E_2, and You consider that at least one holds and its source is fully reliable, but you don't know which one it is.

Let $m^{\Omega}_1 = m^{\Omega}[E_1]$ and $m^{\Omega}_2 = m^{\Omega}[E_2]$ denote the bba induced by E_1 and E_2, respectively.

The belief function that results from the disjunctive combination of m^{Ω}_1 and m^{Ω}_2 can be denoted equivalently

- $m^{\Omega}[E_1 \vee E_2]$ or $m^{\Omega}[E_1 \text{ or } E_2]$
- $m^{\Omega}_1 \text{\textcircled{$\cup$}} m^{\Omega}_2$ or $m^{\Omega}_{1\text{\textcircled{$\cup$}}2}$

where the $\vee$ is the logical 'or', and the $\text{\textcircled{$\cup$}}$ operator denotes the disjunctive combination.

We have

$$m^{\Omega}_{1\text{\textcircled{$\cup$}}2}(C) = \sum_{A\subseteq\Omega, B\subseteq\Omega, A\cup B=C} m^{\Omega}_1(A) m^{\Omega}_2(B), \qquad \forall C \subseteq \Omega \tag{7.24}$$

$$b^{\Omega}_{1\text{\textcircled{$\cup$}}2}(A) = b^{\Omega}_1(A) b^{\Omega}_2(A) \qquad \forall A \subseteq \Omega \tag{7.25}$$

The inverse combination rules

The inverse of the conjunctive combination rule, denoted by $\text{\textcircled{$\bar{\cap}$}}$, is defined as:

$$q^{\Omega}_{1\text{\textcircled{$\bar{\cap}$}}2}(A) = \frac{q^{\Omega}_1(A)}{q^{\Omega}_2(A)}, \qquad \forall A \subseteq \Omega \tag{7.26}$$

provided q^{Ω}_2 is non dogmatic. It satisfies:

$$q^{\Omega}_{(1\text{\textcircled{$\cap$}}2)\text{\textcircled{$\bar{\cap}$}}2} = q^{\Omega}_1 \tag{7.27}$$

The inverse of the Dempster's rule of combination, denoted by $\ominus$, is defined as:

$$q^{\Omega}_{1\ominus 2}(A) = \frac{q^{\Omega}_{1\text{\textcircled{$\bar{\cap}$}}2}(A)}{1 - m^{\Omega}_{1\text{\textcircled{$\bar{\cap}$}}2}(\emptyset)}, \qquad \forall A \subseteq \Omega \tag{7.28}$$

provided q^{Ω}_2 is non dogmatic. It satisfies:

$$q^{\Omega}_{(1\oplus 2)\ominus 2} = q^{\Omega}_1 \tag{7.29}$$

The inverse of the disjunctive combination rule, denoted by $\textcircled{$\uplus$}$, is defined as:

$$b^{\Omega}_{1\textcircled{$\uplus$}2}(A) = \frac{b^{\Omega}_{1}(A)}{b^{\Omega}_{2}(A)}, \qquad \forall A \subseteq \Omega \tag{7.30}$$

provided $b^{\Omega}_{2}(\emptyset) > 0$. It satisfies:

$$b^{\Omega}_{(1\textcircled{$\cup$}2)\textcircled{$\uplus$}2} = b^{\Omega}_{1} \tag{7.31}$$

The negation of a bba

Suppose a piece of evidence E. Let $\sim E$ denote the fact that You consider that the source of the piece of evidence E lies, tells the false, what means that whenever the source says that the actual world belongs to A, You consider it means that it belongs to $\overline{A}$.

Let $m^{\Omega}[E]$ be the bba as produced by the source. You build the bba $m^{\Omega}[\sim E]$, called the negation of the bba $m^{\Omega}[E]$. $m^{\Omega}[\sim E]$ is also denoted as $\overline{m}^{\Omega}$, and it related functions are denoted $\overline{bel}^{\Omega}, \overline{pl}^{\Omega}, \overline{q}^{\Omega}$.

We have:

$$\overline{m}^{\Omega}(A) = m^{\Omega}(\overline{A}), \qquad \forall A \subseteq \Omega \tag{7.32}$$

$$\overline{b}^{\Omega}(A) = q^{\Omega}(\overline{A}), \qquad \forall A \subseteq \Omega \tag{7.33}$$

$$\overline{q}^{\Omega}(A) = b^{\Omega}(\overline{A}), \qquad \forall A \subseteq \Omega \tag{7.34}$$

The De Morgan formula are satisfied.

$$bel^{\Omega}[\sim (E_1 \wedge E_2)] = bel^{\Omega}[\sim E_1 \vee \sim E_2] \tag{7.35}$$

$$bel^{\Omega}[\sim (E_1 \vee E_2)] = bel^{\Omega}[\sim E_1 \wedge \sim E_2] \tag{7.36}$$

or equivalently:

$$\overline{bel^{\Omega}_{1} \textcircled{$\cap$} bel^{\Omega}_{2}} = \overline{bel}^{\Omega}_{1} \textcircled{$\cup$} \overline{bel}^{\Omega}_{2} \tag{7.37}$$

$$\overline{bel^{\Omega}_{1} \textcircled{$\cup$} bel^{\Omega}_{2}} = \overline{bel}^{\Omega}_{1} \textcircled{$\cap$} \overline{bel}^{\Omega}_{2} \tag{7.38}$$

The exclusive disjunctive rule

Suppose two pieces of evidence E_1 and E_2, and You consider that one holds and its source is fully reliable and the other lies, but you don't know which is which.

Let $m^{\Omega}_{1} = m^{\Omega}[E_1]$ and $m^{\Omega}_{2} = m^{\Omega}[E_2]$ denote the bba's induced by E_1 and E_2, respectively.

The belief function that results from the exclusive disjunctive combination of m^{Ω}_{1} and m^{Ω}_{2} can be denoted equivalently

- $m^{\Omega}[E_1 \underline{\vee} E_2]$
- $m^{\Omega}_{1} \underline{\textcircled{$\cup$}} m^{\Omega}_{2}$ or $m^{\Omega}_{1\underline{\textcircled{$\cup$}}2}$

where the $\underline{\vee}$ is the logical 'exclusive or', and the $\underline{\textcircled{$\cup$}}$ operator denotes the exclusive disjunctive combination.

We have

$$m^{\Omega}_{1\underline{\textcircled{$\cup$}}2}(C) = \sum_{A \subseteq \Omega, B \subseteq \Omega, A \underline{\cup} B = C} m^{\Omega}_{1}(A) m^{\Omega}_{2}(B), \qquad \forall C \subseteq \Omega \tag{7.39}$$

where $\underline{\cup}$ denotes the disjoint union.

General combination rule

These three combination rules and the negation rule can be extended to any number of pieces of evidence and any combination formula that states which source You accept as telling the truth.

Let E_1, $E_2 \dots E_n$ be a set of n distinct pieces of evidence produced by the sources $S_1, S_2 \dots S_n$, with $m_i^\Omega = m^\Omega[E_i]$, $i = 1, 2, \dots n$, being the bba's induced by each piece of evidence individually.

For instance, suppose all You accept is that $(E_1 \wedge E_2) \vee E_3)\underline{\vee}(E_4 \wedge E_1)$ holds. It means You accept that one and only one of the two following cases holds: $(E_1 \wedge E_2) \vee E_3$ or $E_4 \wedge E_1$. In the first case, You accept that at least one of the next two cases holds: $(E_1 \wedge E_2)$ or E_3. It means that You accept that either S_1 and S_2 tell the truth or S_3 tells the truth, in a non exclusive way. In the second case, You accept that both S_4 and S_1 tell the truth.

Given this complex piece of evidence, the basic belief assignment related to the bba $m^\Omega[((E_1 \wedge E_2) \vee E_3)\underline{\vee}(E_4 \wedge E_1)]$ is:

$$m^\Omega[((E_1 \wedge E_2) \vee E_3)\underline{\vee}(E_4 \wedge E_1)](A) = \tag{7.40}$$

$$\sum_{X,Y,Z,T \subseteq \Omega, ((X \cap Y) \cup Z) \underline{\cup} (T \cap X) = A} m^\Omega[E_1](X) m^\Omega[E_2](Y) m^\Omega[E_3](Z) m^\Omega[E_4](T) \tag{7.41}$$

for all $A \subseteq \Omega$.

7.4.2 Belief functions induced by non distinct pieces of evidence

Case with known correlation

Consider 6 agents (You, You$_1$, You$_2$, Witness$_1$, Witness$_2$, Witness$_3$). The three witnesses are three distinct sources of evidence on Ω. Let $bel^\Omega_{W_i}$, $i = 1, 2, 3$, be the belief built on Ω by each witness (denoted W_i). Suppose You$_1$ collects the beliefs of W_1 and W_2, and these were his only sources of beliefs over Ω. Let $bel[^\Omega_{Y_1}$ be the belief built by You_1 by conjunctively combining $bel^\Omega_{W_1}$ and $bel^\Omega_{W_2}$, so $bel^\Omega_{You_1} = bel^\Omega_{W_1} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_2}$. Similarly let $bel^\Omega_{You_2} = bel^\Omega_{W_2} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_3}$.

$bel^\Omega_{You_1}$ and $bel^\Omega_{You_2}$ are two belief functions not produced by distinct pieces of evidence, hence the conjunctive combination rule may not be applied to them.

Then You collect the beliefs produced by You$_1$ and You$_2$. If You blindly apply the conjunctive combination rule on $bel^\Omega_{You_1}$ and $bel^\Omega_{You_2}$ as they are not induced by distinct pieces of evidence, what You would get is $bel^\Omega_{W_1} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_2} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_2} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_3}$ whereas You should have computed $bel^\Omega_{W_1} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_2} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_3}$.

Should You know the $bel^\Omega_{W_i}$'s, You would have just conjunctively combined them in order to compute $bel^\Omega_{W_1} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_2} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{W_3}$.

If You knew $bel^\Omega_{W_2}$, You could as well computed

$$bel^\Omega_{You_1} \mathbin{\bigcirc\mkern-15mu\cap} bel^\Omega_{You_2} \mathbin{\bigcirc\mkern-15mu\cap\mkern-12mu-} bel^\Omega_{W_2}.$$

Results are equal.

$bel^\Omega_{W_2}$ is called the 'correlation' between $bel^\Omega_{You_1}$ and $bel^\Omega_{You_2}$.

Cautious combinations

Suppose the much more classical situation where You know there is some evidence that has been used by both You_1 and You_2, but You don't know which one.

All You know is that Your belief bel^{Ω}_{You} must result from the conjunctive combination of both $bel^{\Omega}_{You_1}$ and $bel^{\Omega}_{You_2}$ with some unknown belief functions. Let $\mathcal{B}_i$ be the set of belief functions that can be build from a conjunctive combination of $bel^{\Omega}_{You_i}$ with any belief function on Ω.

$$\mathcal{B}_i = \{bel^{\Omega} : \exists bel^{\Omega}_0, bel^{\Omega} = bel^{\Omega}_{You_i} \textcircled{\scriptsize $\cap$} bel^{\Omega}_0\}$$

Then $bel^{\Omega}_{You} \in \mathcal{B}_1$ and $bel^{\Omega}_{You} \in \mathcal{B}_2$, hence $bel^{\Omega}_{You} \in \mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2$.

Assume the solution, denoted $bel^{\Omega}_{You_1} \owedge bel^{\Omega}_{You_2}$, has to be

- idempotent: if $bel^{\Omega}_{You_1} = bel^{\Omega}_{You_2}$, then $bel^{\Omega}_{You} = bel^{\Omega}_{You_1}$, i.e., $bel^{\Omega} \owedge bel^{\Omega} = bel^{\Omega}$,
- commutative: permuting $bel^{\Omega}_{You_1}$ and $bel^{\Omega}_{You_2}$ does not affect the result, i.e., $bel^{\Omega}_1 \owedge bel^{\Omega}_2 = bel^{\Omega}_2 \owedge bel^{\Omega}_1$,
- associative: the order with which the belief functions are combined does not affect the result, i.e., $(bel^{\Omega}_1 \owedge bel^{\Omega}_2) \owedge bel^{\Omega}_3 = bel^{\Omega}_1 \owedge (bel^{\Omega}_2 \owedge bel^{\Omega}_3)$,

For non dogmatic belief functions, the solution is unique.

This rule is called the cautious conjunctive combination rule. Bold disjunctive combination rule exist also but are not examined here.

7.5 The Principle of Minimal Commitment

7.5.1 The pl-LC

Let Me (or I) be another agent, different from You. Suppose You only know that My belief function over $\Omega = \{a, b, c\}$ is such that $bel_{\mathrm{Me}}(\{a\}) = .3$ and $bel_{\mathrm{Me}}(\{b, c\}) = .5$, and You do not know the value given to bel_{Me} for the other subsets of Ω. Suppose You have no other information on Ω and You are ready to adopt My belief as Yours. How to build Your beliefs given these partial constraints? Many belief functions can satisfy them. If you adopt the principle that subsets of Ω should not get more support than justified, then Your belief on Ω will be such that $m_{\mathrm{You}}(\{a\}) = .3$, $m_{\mathrm{You}}(\{b, c\}) = .5$ and $m_{\mathrm{You}}(\{a, b, c\}) = .2$. Among all belief functions compatible with the constraints given by known values of bel_{Me}, bel_{You} is the one that gives the smallest degree of belief to every subset of Ω. The principle evokes here is called the **Principle of pl-Minimal Commitment**. It is really at the core of the TBM, where degrees of belief are degrees of 'justified' supports.

With un-normalized belief functions (where $m(\emptyset)$ can be positive), the definition of the principle is based on the plausibility functions. Suppose two plausibility functions pl_1 and pl_2 such

$$pl_1(A) \leq pl_2(A) \qquad \forall A \subseteq \Omega.$$

We say that pl_2 is not more committed than pl_1 (and less committed if there is at least one strict inequality). The same qualification is extended to their related basic belief assignments and belief functions. Among all belief functions on Ω, the pl-least committed belief function is the vacuous belief function where $pl(A) = 1$ for all $A \neq \emptyset$.

When expressed with belief functions, the principle becomes:

$$\begin{array}{lll} & b_1(A) \geq b_2(A) & \forall A \subseteq \Omega \\ \text{i.e.,} & bel_1(A) + m_1(\emptyset) \geq bel_2(A) + m_2(\emptyset) & \forall A \subseteq \Omega. \end{array}$$

The concept of 'pl-least commitment' permits the construction of a partial order $\sqsubseteq$ on the set of belief functions.

We write:

$$pl_1 \sqsubseteq_{pl} pl_2$$

to denote that pl_1 is equal to or more committed than pl_2. By analogy the following notations $m_1 \sqsubseteq_{pl} m_2$ and $bel_1 \sqsubseteq_{pl} bel_2$ have the same meaning.

The **Principle of pl-Minimal Commitment** consists in selecting the least committed belief function in a set of equally justified belief functions. The principle formalizes the idea that one should never give more support than justified to any subset of Ω. It satisfies a form of skepticism, of uncommitment, of conservatism in the allocation of our belief. In its spirit, it is not far from what the probabilists try to achieve with the maximum entropy principle.

This principle does not work in every situation. The set of belief functions compatible with a given set of constraints does not always admit a unique least committed solution. In that case extra requirements must be introduced, like those based on the information content of a belief function.

7.5.2 Other orders

One could also defend that the least commitment should be based on the q function, hence the q-LC principle:

$$m_1 \sqsubseteq_q m_2 \text{ iff } q_1(X) \geq q_2(X), \ \forall X \subseteq \Omega$$

Similarly an order could be based on the specialization relation:

$$m_1 \sqsubseteq_{sp} m_2 \text{ iff } m_2 \text{ is a specialization of } m_1, \ \forall X \subseteq \Omega$$

One has

$$\sqsubseteq_{sp} \Rightarrow \sqsubseteq_{pl} \ \& \sqsubseteq_q$$

but not the reverse, and one proves that $\sqsubseteq_{pl}$ and $\sqsubseteq_q$ are not equivalent.

7.6 The Generalized Bayesian Theorem

Suppose the finite spaces X and Θ. Suppose that for each $\theta \in \Theta$, there is a basic belief assignment on X, denoted $m^X[\theta]$. Given this set of basic belief assignments, what is the belief induced on Ω if You come to know that $x \subseteq X$ holds?

In statistics, this is the classical inference problem where Θ is a parameter space and X an observation space. You know the probability functions on X given each θ in Θ, You observe $x \subseteq X$ and You compute, by applying the bayesian theorem, the probability function on Θ given x, using also an a priori probability function on Θ.

The Generalized Bayesian Theorem (GBT) performs the same task with belief functions. The major point is that the needed prior can be a vacuous belief function, what is the perfect representation of total ignorance. No informative prior belief is needed, avoiding thus one of the major criticisms against the classical approach, in particular when used for diagnostic applications.

Given the set of basic belief assignments $m^X[\theta]$ known for every $\theta \in \Theta$ and their related b-functions $b^X[\theta]$ and plausibility functions $pl^X[\theta]$, then for $x \subseteq X$ and for every $A \subseteq \Theta$:

$$\begin{aligned}
b^\Theta[x](A) &= \prod_{\theta \in \overline{A}} b^X[\theta](X - x) \\
pl^\Theta[x](A) &= 1 - \prod_{\theta \in A} (1 - pl^X[\theta](x)) \\
q^\Theta[x](A) &= \prod_{\theta \in A} pl^X[\theta](x)
\end{aligned}$$

where $[x]$ is the piece of evidence that states ‘x holds’.

Should You have some non vacuous beliefs on Θ, represented by $m^\Theta[E_0]$, than this belief is simply combined with $m^\Theta[x](A)$ by the application of the conjunctive rule of combination.

This rule has been derived axiomatically by Smets (1978, 1993) and by Appriou (1991).

Some particular cases are worth mentioning.

1. Independent events We consider the case of two ‘independent’ observations x defined on X and y defined on Y, and the inference on Θ obtained from their joint observation.

Suppose the two variables X and Y satisfy the **Conditional Cognitive Independence** property defined as:

$$pl^{X \times Y}[\theta](x, y) = pl^X[\theta](x) pl^Y[\theta](y) \; \forall x \subseteq X, \forall y \subseteq Y, \forall \theta_i \in \Theta.$$

This property is the generalization of the classical independence described in probability theory. It reduces to the classical conditional independence property when all plausibility functions are probability functions.

Let $b^\Theta[x]$ and $b^\Theta[y]$ be computed by the GBT (with a vacuous a priori belief on Θ) from the set of basic belief assignment $m^X[\theta]$ and $m^Y[\theta]$ known for every $\theta \in \Theta$. We then combine by the conjunctive rule of combination these two functions in order to build the belief $b^\Theta[x, y]$ on Θ induced by the pair of observations.

We could as well consider the basic belief assignment $m^{X \times Y}[\theta]$ built on the space $X \times Y$ thanks to the Conditional Cognitive Independence property, and compute $b^\Theta[x, y]$ from it using the GBT. Both results are the same, a property that is essential and at the core of the axiomatic derivation of the rule.

2. Degradation with probabilities. If for each $\theta \in \Theta$, $b^X[\theta]$ is a probability function $P(\dot{|}\theta)$ on X, then the GBT for $|\theta| = 1$ becomes:

$$pl^\Theta[x](\theta) = P(x|\theta) \qquad \forall x \subseteq X$$

That is, on the singletons θ of Θ, $pl^{\Theta}[x]$ reduces to the likelihood of θ given x. The analogy stops there as the solution for the likelihood of subsets of Θ are different.

If, furthermore, the *a priori* belief on θ is also a probability function $P_0(\theta)$, then the normalized GBT becomes:

$$bel^{\Theta}[x](A) = \frac{\sum_{\theta \in A} P(x|\theta)P_0(\theta)}{\sum_{\theta \in \Theta} P(x|\theta)P_0(\theta)} = P(A|x)$$

i.e. the (normalized) GBT reduces itself into the classical Bayesian theorem, which explains the origin of its name.

7.7 Decision Making.

7.7.1 The pignistic probability function for decision making

Suppose a frame of discernment Ω with *bel* quantifying Your beliefs on Ω at the credal level. When a decision must be made that depends on the actual state of affairs ω_0 where $\omega_0 \in \Omega$, You must construct a probability function on 2^{Ω} in order to make the optimal decision, i.e., the one that maximizes the expected utility. If You did not, Your behavior could be shown to be 'irrational'. We assume that the probability function defined on 2^{Ω} is a function of the belief function *bel* also defined on 2^{Ω}. It translates the saying that beliefs guide our actions. Hence one must transform *bel* into a probability function, denoted $BetP$. This transformation is called the pignistic transformation and denoted Γ, that depends on both *bel* and the betting frame F on which bets are built.

7.7.2 The betting frame

The pignistic transformation depends on the structure of the frame on which decision must be made. The betting frame F on Ω is the set of 'atoms' on which stakes will be allocated. Bets can then be built only on the elements of the power set of that frame. The granularity of the frame F is defined so that a stake could be given to each atom of F independently of the stakes given to the other atoms of F. Suppose one starts with a credibility function on a frame F_0. If the stakes given to atoms A and B of F_0 must necessarily be always equal, both A and B belong to the same granule of the betting frame F. The betting frame is organized so that its granules are the atoms of F. F results from the application of a sequence of coarsenings and/or refinements on F_0. The pignistic probability $BetP$ is then built from the belief function bel so derived on F, and we write:

$$BetP = \Gamma(bel, F).$$

We call $BetP$ a pignistic probability to insist on the fact that it is a probability measure used to make decisions (Bet is for betting). Of course $BetP$ is a classical probability measure.

7.7.3 The pignistic transformation

Let m be the basic belief assignment on 2^F related to the belief function bel also defined on 2^F. Let $BetP = \Gamma(bel, F)$. Then:

$$BetP(\omega) = \sum_{A:\omega\in A\in 2^F} \frac{m(A)}{|A|(1-m(\emptyset))},\ \forall\omega\in F, \tag{7.42}$$

where $|A|$ is the number of elements of F in A.

It is easy to show that the function $BetP$ is a probability function and the pignistic transformation of a probability function is the probability function itself.

7.8 The canonical decompositions

For $A \subseteq \Omega$ and $wc_A \in [0,\infty)$, let A^{wc_A} denote a generalized simple support function, i.e. a 'basic belief assignment' which 'masses' are given by: $m(\Omega) = wc_A$, $m(A) = 1 - wc_A$, the other m's being 0. The generalized' comes from the fact the masses may be negative (when $x_A > 1$).

Any non dogmatic bba $m\Omega$ can be represented as:

$$m^\Omega = \bigcirc\!\!\!\!\!\cap_{A\subseteq\Omega} A^{wc_A}$$

where the wc_A are the canonical weights defined in section (7.2.5) and the $\bigcirc\!\!\!\!\!\cap$ operator is extended to the whole R domain.

This canonical representation satisfies: If wc^1_A and wc^2_A are the conjunctive canonical weights of m^Ω_1 and m^Ω_2, then $wc^1_A.wc^2_A$ are the conjunctive canonical weights of $m^\Omega_1 \bigcirc\!\!\!\!\!\cap m^\Omega_2$.

There exists also a similar disjunctive decomposition.

For $A \subseteq \Omega$ and $wd_A \in [0,\infty)$, let A_{wd_A} denote a 'basic belief assignment' which 'masses are given by: $m(\emptyset) = wd_A, m(A) = 1 - wd_A$, the other m's being 0.

Any non-conflicting bba m^Ω can be represented as:

$$m^\Omega = \bigcirc\!\!\!\!\!\cup_{A\subseteq\Omega} A_{wd_A}$$

where the wd_A are the disjunctive canonical weights defined in section (7.2.5) and the $\bigcirc\!\!\!\!\!\cup$ operator is extended to the whole R domain.

This canonical representation satisfies: If wd^1_A and wd^2_A are the disjunctive canonical weights of m^Ω_1 and m^Ω_2, then $wd^1_A.wd^2_A$ are the disjunctive canonical weights of $m^\Omega_1 \bigcirc\!\!\!\!\!\cup m^\Omega_2$.

7.9 The Valuation Based Systems

The VBS is an algorithm used for propagating beliefs (and probabilities) in networks. The nodes of the networks are variables, and the links between sets of nodes are joint belief functions defined on of the product space of the variables described by the nodes related by the link. User is referred to Shenoy bibliography for detailed information about the VBS.

Initially the VBS considered only joint belief functions, but it can be extended to conditional belief functions.

7.10 Belief functions on continuous spaces

The topic is not yet finalized nor published. So this section will shop up once the theory will be finalized.

References

Abramowitz, M., & Stegun, I. A. (1965). *Handbook of mathematical functions.* Dover Publications, Inc., N.Y.

Dempster, A. P. (1966). New methods for reasoning towards posterior distributions based on sample data. *Ann. Math. Statistics*, *37*, 355–374.

Dempster, A. P. (1967a). Upper and lower probabilities induced by a multiple valued mapping. *Ann. Math. Statistics*, *38*, 325–339.

Dempster, A. P. (1967b). Upper and lower probability inferences based on a sample from a finite univariate population. *Biometrika*, *54*, 515–528.

Dempster, A. P. (1968a). Upper and lower probabilities generated by a random closed interval. *Ann. Math. Statistics*, *39*, 957–966.

Dempster, A. P. (1968b). A generalization of Bayesian inference. *J. Roy. Statist. Soc. B.*, *30*, 205–247.

Dempster, A. P. (1969). Upper and lower probability inferences for families of hypothesis with monotone density ratios. *Ann. Math. Statistics*, *40*, 953–969.

Dempster, A. P. (1972). A class of random convex polytopes. *Annals of Mathematical Statistics*, *43*, 250–272.

Kohlas, J., & Monney, P. A. (1995). *A mathematical theory of hints: An approach to Dempster-Shafer theory of evidence.* Lecture Notes in Economics and Mathematical Systems No. 425. Springer-Verlag.

Shafer, G. (1976). *A mathematical theory of evidence.* Princeton Univ. Press. Princeton, NJ.

Smets, P. (1998). The transferable belief model for quantified belief representation. In D. M. Gabbay & P. Smets (Eds.), *Handbook of defeasible reasoning and uncertainty management systems* (Vol. 1, pp. 267–301). Kluwer, Doordrecht, The Netherlands.

Smets, P., & Kennes, R. (1994). The transferable belief model. *Artificial Intelligence*, *66*, 191–234.

Chapter 8

Indexes

Chapter 9

Software License

This library is free software; you can redistribute it and/or modify it under the following terms.

Redistribution and use of source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
- Any use of the present software or part of it for commercial or profit purposes is forbidden without specific prior written permission by the copyright holder.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.