Belief functions, propositional logic and Boolean structures: some computational and practical aspects

Sébastien Destercke

Heudiasyc, CNRS Compiegne, France

BFAS 2019 School

Lecture plan

What you will find in this talk

- Some (sketchy) reminders about Propositional logic
- Combining belief functions and Boolean logic
- Computational aspects of Boolean inferences
- Some applications

Plan

An introductory story

You are organizing a party in some (obviously) fictitious world, and have invited Arthur Dempster (A), Bolsonaro (B) and Claude Shannon (C). You know:

- That if Arthur is coming, then Bolsonaro (as a big belief function fan) will want to come

$$a \to b$$

- Bolsonaro and Claude do not get along, they will not come

$$\neg(b \wedge c)$$

You want to model these pieces of knowledge, and reason with them.

Formally speaking

Some elements of the propositional language we consider:

- A set of n **Boolean variables** $X_i \in \{x_i, \neg x_i\}$, $i \in \{1, \ldots, n\}$

  In examples: alphabetic letters a, b, c

- A **formula** ϕ is a composition of connectors $\{\neg, \wedge, \vee\}$ and variable values/other formulas, possibly in a compact form.

  For instance: $a \rightarrow b : \neg a \vee b$

- An **interpretation** ω is an assignment of truth values to variables, and $\Omega = \times_{i=1}^{n} \{x_i, \neg x_i\}$ is the set of all interpretations
- The **models** $E_\phi \subseteq \Omega$ of ϕ are the interpretations satisfying ϕ (for which ϕ is true)

  Ex: $E_{\neg(b \wedge c)} = \{ab\neg c, a\neg bc, a\neg b\neg c, \neg ab\neg c, \neg a\neg bc, \neg a\neg b\neg c\}$

- We will say that a set of formula $\phi_1 \wedge \ldots \wedge \phi_n$ form a knowledge basis

Logic and belief functions: a first glimpse

- A formula ϕ is semantically equivalent to its set E_ϕ of models: we will interchangeably use them
- The set of interpretations $\Omega \equiv$ frame of discernment. This means 2^n atoms/singletons.
- The models E_ϕ of a formula $\equiv$ a focal elements/an event
- $\phi_1 \wedge \phi_2 \equiv E_{\phi_1} \cap E_{\phi_2}$
- A formula without uncertainty $\equiv$ a categorical BF
- $E_\phi = \emptyset$ if ϕ inconsistent knowledge

The story continued

- Bolsonaro will come if Arthur comes.

$$E_{a\to b} = \{abc, ab\neg c, \neg abc, \neg a\neg bc, \neg ab\neg c, \neg a\neg b\neg c\}$$

- Claude and Bolsonaro cannot be together

$$E_{\neg(b\wedge c)} = \{ab\neg c, a\neg bc, a\neg b\neg c, \neg ab\neg c, \neg a\neg bc, \neg a\neg b\neg c\}$$

$$\to m(E_{\neg(b\wedge c)} \cap E_{a\to b}) = 1$$

- Question (another formula): is Bolsonaro coming? Is b true?

$$A_b = \{abc, \neg abc, ab\neg c, \neg ab\neg c\}$$

$$Bel(A_b) = 0, Pl(A_b) = 1 \text{ (We do not know)}$$

The story continued

- Bolsonaro will come if Arthur comes.

$$E_{a \to b} = \{abc, ab\neg c, \neg abc, \neg a \neg bc, \neg ab \neg c, \neg a \neg b \neg c\}$$

- Claude and Bolsonaro cannot be together

$$E_{\neg(b \wedge c)} = \{ab\neg c, a\neg bc, a\neg b\neg c, \neg ab\neg c, \neg a\neg bc, \neg a\neg b\neg c\}$$

$$\to m(E_{\neg(b \wedge c)} \cap E_{a \to b}) = 1$$

- Question (another formula): are Arthur and Claude coming?

$$A_{\{a \cap c\}} = \{abc, a\neg bc\}$$

$$Bel(A_{\{a \cap c\}}) = 0, Pl(A_{\{a \cap c\}}) = 0 \text{ (No, they will not)}$$

Plan

Outline

Correspondences

- m represents uncertainty knowledge over formulas
- focal sets $\equiv$ conjunction of formulas
- Bel: probability that A is implied by our knowledge
- Pl: probability that A is consistent with our knowledge
- Dempster's rule: product of masses + conjunction of rules

The example revisited

- Assume there are extreme rare cases where both Bolsonaro and Claude were present
- The information $\neg(b \wedge c)$ is not certain anymore ($\exists$ counter-examples).
- Most of the time, they are not together:

$$Bel(E_{\neg(b \wedge c)}) = 0.9$$

$$\Downarrow$$

$$m(E_{\neg(b \wedge c)}) = 0.9, m(\Omega) = 0.1$$

  Simple support function
- Similarly, it is only very likely that B. come if A. comes

$$m(E_{a \rightarrow b}) = 0.8, m(\Omega) = 0.2$$

The example continued

- our final mass function, once the two are combined is

$$m(E_{\neg(b\wedge c)} \cap E_{a\rightarrow b}) = 0.72$$

$$m(E_{\neg(b\wedge c)}) = 0.18$$

$$m(E_{a\rightarrow b}) = 0.08$$

$$m(\Omega) = 0.02$$

- event $b \wedge c$ no longer impossible:

$$Pl(A_{b\wedge c}) = m(E_{a\rightarrow b}) + m(\Omega) = 0.1$$

but still quite unlikely.

Other combination rules

- One can of course combine masses and logical formulas in other ways, using other rules
- If masses are simple support functions

$$m_i(E_{\phi_i}) = \alpha_i, m_i(\Omega) = 1 - \alpha_i$$

then we can retrieve possibilistic logic [9] by using the equivalent of the minimum combination rule (see [8] for a justification/study of it within DS theory).

Incomplete note on interpretations and other works

Here, focus on purely formal/computational extensions. Other authors have explored the semantic/interpretation side of combining belief functions ideas with (modal) logic:

- "A simple logic for reasoning about incomplete knowledge.", M. Banerjee, D. Dubois [1]
- "Towards a logical belief function theory.", L. Cholvy [4]
- "Penalty logic and its link with Dempster-Shafer theory", F.D. De Saint-Cyr, J. Lang, T. Schiex [7]
- ...

also, DS theory often appears as a nice way to combine probability and logic, as logical inferences often ends up with sets of models, such as in

- "The Joy of Probabilistic Answer Set Programming", F. Cozman [6]

Outline

Conflict and belief functions

- $m(\emptyset)$= sum of mass given to inconsistent knowledge basis/set of formulas
- For a given logical basis $K = \phi_1 \cap \ldots \cap \phi_n$, this amounts to "plug in" for each focal elements an inconsistency measure *Inc* that is the following:

$$Inc(K) = \begin{cases} 1 & \text{if } K \text{ inconsistent} \\ 0 & \text{else} \end{cases}$$

- One may then question whether this is a good inconsistency measure?

An example

Consider two variables a, b and $\Omega = \{ab, \neg ab, a\neg b, \neg a\neg b\}$

$$m_1(E_{a\wedge b}) = \alpha_1 \quad m_1(\Omega) = 1 - \alpha_1$$

$$m_2(E_{\neg a}) = \alpha_2 \quad m_2(\Omega) = 1 - \alpha_2$$

$\Downarrow$ Dempster

$$m_{12}(\emptyset) = ?$$

$$m_1(E_{a\wedge b}) = \beta_1 \quad m_1(\Omega) = 1 - \beta_1$$

$$m_2(E_{\neg a}) = \beta_2 \quad m_2(\Omega) = 1 - \beta_2$$

$$m_3(E_{\neg b}) = \beta_3 \quad m_3(\Omega) = 1 - \beta_3$$

$\Downarrow$ Dempster

$$m_{123}(\emptyset) = ?$$

An example

Consider two variables a, b and $\Omega = \{ab, \neg ab, a\neg b, \neg a\neg b\}$

$$m_1(E_{a\wedge b}) = \alpha_1 \quad m_1(\Omega) = 1 - \alpha_1$$

$$m_2(E_{\neg a}) = \alpha_2 \quad m_2(\Omega) = 1 - \alpha_2$$

$\Downarrow$ Dempster

$$m_{12}(\emptyset) = \alpha_1\alpha_2$$

$$m_1(E_{a\wedge b}) = \beta_1 \quad m_1(\Omega) = 1 - \beta_1$$

$$m_2(E_{\neg a}) = \beta_2 \quad m_2(\Omega) = 1 - \beta_2$$

$$m_3(E_{\neg b}) = \beta_3 \quad m_3(\Omega) = 1 - \beta_3$$

$\Downarrow$ Dempster

$$m_{123}(\emptyset) = \beta_1\beta_2 + \beta_1(1 - \beta_2)\beta_3$$

Hence, $m_{123}(\emptyset) = m_{12}(\emptyset)$ iff $\alpha_1\alpha_2 = \beta_1\beta_2 + \beta_1(1 - \beta_2)\beta_3$

Question: are those two situations equivalent conflict-wise?

An example, continued

However, in the two cases the conflict comes from different situations:

- In the first case, from

$$m_{12}(\{a \wedge b\} \wedge \{\neg a\})$$

- In the second case, from

$$m_{123}(\{a \wedge b\} \wedge \{\neg a\}) + m_{123}(\{a \wedge b\} \wedge \{\neg b\}) + m_{123}(\{a \wedge b\} \wedge \{\neg a\} \wedge \{\neg b\})$$

- Most inconsistency measures used in logic would agree that the situations are different [10]
- Idea to be explored: replace $Inc(K)$ by other inconsistency measures issued from logic literature?

Conflict and inconsistency, continued

For instance, consider the bases:

- $K_1 = \{a \wedge b\}, \{\neg a\}, \{b\}$
- $K_2 = \{a \wedge b\}, \{\neg a\}, \{\neg b\}$

Most inconsistency indices from logic would consider K_2 more inconsistent than K_1

$\Rightarrow$ any subsets of size more than one in K_2 is inconsistent.

Links and mixes between the two fields constitute a possible interesting line of research., e.g., to better differentiate different conflicting situations, or identify sources of conflicts.

Plan

Outline

Some simple observations

- If we have n variables X_i, we have 2^n possible world states, atoms, ... in Ω
- A generic belief function would have at most...

$$2^{2^n}$$

  focal elements. For $n = 10$, this is a number with more than 300 digits!
- Clearly, you either have to
  - find another research topic;
  - wait for quantum computing;
  - work with approximations or specific cases that allow for efficient computations, and be clever to find those.

A first solution: using modern tools

- A focal set $E \equiv$ a set K of (conjunctions of) logical formulas
- Checking if $E = \emptyset$

  $\rightarrow K$ satisfiable? [1]

- Checking if $E \subseteq A$ (is A implied by E?)

  $\rightarrow K \wedge \neg A$ satisfiable? [2]

- Checking if $E \cap A \neq \emptyset$ (is A consistent with E)

  $\rightarrow K \wedge A$ satisfiable?

All of this correspond to solve SAT problems.

[1] K satisfiable iff $\exists$ model of K, i.e., an assignment of truth values on variables for which K is true.

[2] Remember that set of models and formulas are in 1-to-1 correspondence.

Are we there yet?

- Bad news: Solving SAT problems is in general NP-Hard!
- Good news:
  - As usual, this is a worst complexity result, not an average one
  - Very efficient SAT solvers are now available
  - Many subcases are easier to solves
    - 2-SAT
    - Horn clauses (disjunction with 0 or 1 positive literal)
    - Clauses written with XOR
- In the next step, we are not considering them, but rather restrict the belief functions and events we will consider to specific cases.

If this is so hard, why bother?

Propositional languages and SAT used in plenty of domains:

- Knowledge representation, including ontologies, grounded first order logics, . . . but not only!
- Preference representation
- Solving combinatorial problems of all kinds (e.g., CSP seen as SAT instances)
- Planning, model checking, . . .

Adding a rich uncertainty language to them could be interesting to explore, just for fun or as they may turn out to important in various applications.

Outline

Where is the uncertainty?

- We consider the n Boolean variables X_i.
- We consider that (normalized) uncertainty bears only on each variable X_i:

$$m_i : 2^{\{x_i, \neg x_i\}} \to [0, 1]$$

  inducing

$$[Bel(x_i), Pl(x_i)] \text{ and } [Bel(\neg x_i) = 1 - Pl(x_i), Pl(\neg x_i) = 1 - Bel(x_i)]$$

  hence one interval contains all information.
- We will also assume some kind of independence between those models

Boolean function and multi-linearity

- Any Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ can be mapped into a multi-linear formula (associating x_i to 1, $\neg x_i$ to 0)
- An event A (or an associated formula) is such a function
- Assume job is done, for instance using BDD [2, 3]
- We have a disjunction of m disjoint terms $A_i^{(P_i,N_i)}$ where
  - $P_i \subseteq \{1, \dots, n\}$ is the set of positive literals/variables
  - $N_i \subseteq \{1, \dots, n\}$ is the set of negative literals/variables

  Then

$$f(x_1, \dots, x_n) = \sum_{i=1}^{n} \prod_{j \in P_i} x_j \prod_{j \in N_i} (1 - x_j)$$

  which is a sum of products of degree 1 at most

A very small, illustrative example

- Two pumps X_1 and X_2 that can be functioning (x_i) or not ($\neg x_i$)
- System works if one and only one of them works (XOR connector):
  - If two pumps work, overloading
  - If none works, no pumping

$$A = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$$

encoded by

$$f(x_1, x_2) = x_1(1 - x_2) + (1 - x_1)x_2$$

The probabilistic case

- Each variable X_i has probability $p(x_i)$ of being true, independently of other variables
- Estimate $P(A)$ under these conditions?
- Simple: replace x_i by $p(x_i)$ in $f(x_i)$.
- In our previous example:

$$\begin{aligned} P(A) &= p(x_1)(1 - p(x_2)) + (1 - p(x_1))p(x_2) \\ &= p(x_1)p(\neg x_2) + p(\neg x_1)p(x_2) \end{aligned}$$

- If $p(x_1) = 0.9$ and $p(x_2) = 0.8$, then

$$P(A) = 0.9 \cdot 0.2 + 0.1 \cdot 0.8 = 0.26$$

The belief case (IP style)

- Each variable X_i uncertainty is now characterised by a mass function, and we have

$$[Bel(x_i), Pl(x_i)]$$

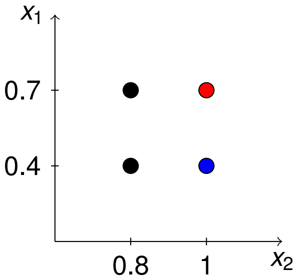
- Assuming[3] that $p(x_i) \in [Bel(x_i), Pl(x_i)]$, compute the bounds $[\underline{P}(A), \overline{P}(A)]$
- In the general case, as $P(A)$ is a multi-linear form of $p(x_i)$, it is monotonous in each $p(x_i)$ when all others are fixed.
- This means that the bounds $[\underline{P}(A), \overline{P}(A)]$[4] are obtained on the vertices of $\times_{i=1}^{n}[Bel(x_i), Pl(x_i)]$, i.e. in $\times_{i=1}^{n}\{Bel(x_i), Pl(x_i)\}$

[3] We consider an imprecise probabilistic interpretation called strong independence [5]. More about a DS interpretation later on.

[4] Not belief functions in general.

The belief case on a simple example

- $[Bel(x_1), Pl(x_1)] = [0.4, 0.7]$
- $[Bel(x_2), Pl(x_2)] = [0.8, 1]$



$$P(A) = p(x_1)p(\neg x_2) + p(\neg x_1)p(x_2)$$

$$\underline{P}(A) = 0.7 \cdot 0 + 0.3 \cdot 1 = 0.3$$

$$\overline{P}(A) = 0.4 \cdot 0 + 0.6 \cdot 1 = 0.6$$

Is it the end?

- Simple strategy to find answers: check every vertex
- Yeah, but. . . do you see the catch?
- Can we do better than checking every one of them?
- In general, no $\rightarrow$ still a difficult problem
- Also, what about other DS interpretation, i.e., in terms of products of masses

Previous case - revisited

$$[Bel(x_1), Pl(x_1)] = [0.4, 0.7] \qquad [Bel(x_2), Pl(x_2)] = [0.8, 1]$$

$$m(x_1) = 0.4$$
$$m(\neg x_1) = 0.3$$
$$m(x_1, \neg x_1) = 0.3$$

$$m(x_2) = 0.8$$
$$m(\neg x_2) = 0$$
$$m(x_2, \neg x_2) = 0.2$$

$\Downarrow$ Ballooning + Dempster[5]

$$m(x_1 \times x_2) = 0.32, \quad m(x_1 \times \{x_2, \neg x_2\}) = 0.08$$
$$m(\neg x_1 \times x_2) = 0.24, \quad m(\neg x_1 \times \{x_2, \neg x_2\}) = 0.06$$
$$m(\{x_1, \neg x_1\} \times x_2) = 0.24, \quad m(\{x_1, \neg x_1\} \times \{x_2, \neg x_2\}) = 0.06$$

[5] Also named random set independence [5]

Previous case - revisited 2

$$m(x_1 \times x_2) = 0.32, \quad m(x_1 \times \{x_2, \neg x_2\}) = 0.08$$
$$m(\neg x_1 \times x_2) = 0.24, \quad m(\neg x_1 \times \{x_2, \neg x_2\}) = 0.06$$
$$m(\{x_1, \neg x_1\} \times x_2) = 0.24, \quad m(\{x_1, \neg x_1\} \times \{x_2, \neg x_2\}) = 0.06$$
$$\Downarrow$$
$$Bel((x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)) = 0.24 < 0.3$$
$$Pl((x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)) = 0.68 > 0.6$$

$\Rightarrow$ always includes previous approach

An interesting subcase

- Assume your function/event is globally monotone in each x_i
- This means $f(\dots, \neg x_i, \dots) \leq f(\dots, x_i, \dots)$, I.e., a variable becoming true cannot make the function f go from true to false.
- Two interesting results in this case:
  - bounds $\underline{P}(A), \overline{P}(A)$ are obtained on the vertices $\times_{i=1}^{n} Bel(x_i)$ and $\times_{i=1}^{n} Pl(x_i)$, respectively $\Rightarrow$ huge computational gain
  - the two approaches, probabilistic bounds and DS, coincide $\Rightarrow$ no need to choose one or the other
- In the next part, we explore some cases where such monotonicity occurs, and is taken advantage of.

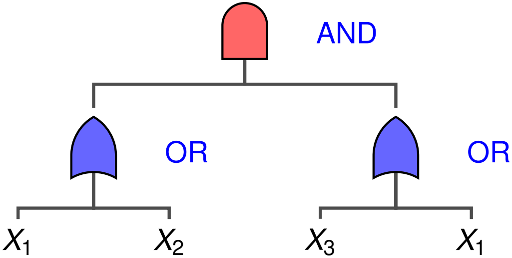
Plan

Outline

The framework

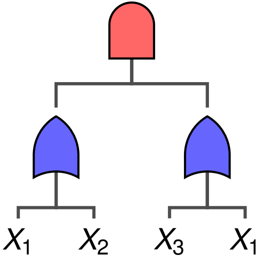
- X_i are component, and a component can be working (x_i) or not working ($\neg x_i$)
- Usually encoded as 1 (x_i) and 0 ($\neg x_i$)
- System working=Boolean function $f : \{0,1\}^n \to \{0,1\}$ describing states in which system works, corresponding to event A that systems works
- Monotone function: if a component goes from failing ($X_i = 0$) to working ($X_i' = 1$), then $f(X) \leq f(X')$

Fault trees

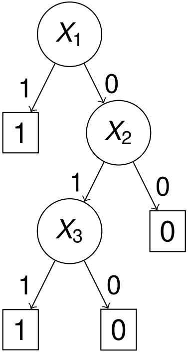


$$A = (x_1 \vee x_2) \wedge (x_3 \vee x_1)$$

From fault trees to BDD to easy formulas



$\Rightarrow$



$$A = (x_1) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

Computational advantage: union of disjoint events

Applying the previous results

$$A = (x_1) \vee (\neg x_1 \wedge x_2 \wedge x_3)$$

$$P(A) = P(x_1) + P(\neg x_1 \wedge x_2 \wedge x_3)$$

If information on atomic events, then

$$\underline{P}(A) = Bel(x_1) + Pl(\neg x_1) \cdot Bel(x_2) \cdot Bel(x_3)$$

To compare to the (very crude) approximation

$$P_*(A) = Bel(x_1) + Bel(\neg x_1) \cdot Bel(x_2) \cdot Bel(x_3)$$

that ignores the redundancy of x_1 in both terms.

Applying numbers

If we have

$$[Bel(x_1), Pl(x_1)] = [0, 1]$$

$$[Bel(x_2), Pl(x_2)] = [0.8, 0.9]$$

$$[Bel(x_3), Pl(x_3)] = [0.7, 0.8]$$

Then

$$\underline{P}(A) = 0 + 1 \cdot 0.7 \cdot 0.8 = 0.56$$

$$P_*(A) = 0$$

Reliability and imprecision: some comments

- Any (coherent) reliability function satisfies monotonicity, meaning that we can use our results
- This also holds for cases where the states of each components are non-binary but ordered (multi-state problems)

Reliability and imprecision: some research questions

- Not all syntactic decompositions of a formula have the same complexity $\rightarrow$ Is picking the most compact/computationally efficient for monotonic functions easy?
- How to design systems, or eliminate non-optimal ones in an imprecise setting (using, e.g., decision rules from Thierry's talk)? $f_1 - f_2$ to compare systems 1 and 2 is no longer monotonic.
- If we can do additional experiments, on which component should we do them?

Outline

The problem

To direct itself, a vehicle/robot/moving object must

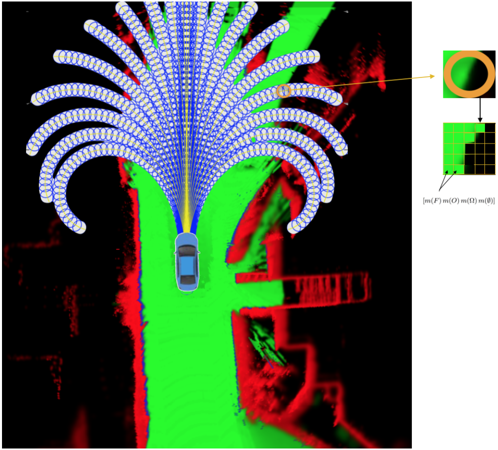
- Perceive its environment
- Plan its trajectory accordingly

This is not without issues as

- Perception is uncertain (occlusion, sensor noise)
- Planning must sometimes be done in real time

One can use our previous results to (tentatively) solve both issues

An illustration



Mouhagir et al., IFAC, 2017

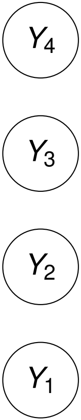
- map= set of thousand cells X_i that can be occupied (x_i) or not ($\neg x_i$)
- 1 trajectory = set of occupied positions Y_j (circles)
- 1 position Y_j= a set of elementary cells X_i

Question: which trajectory should we pick?

A possible solution

For a trajectory

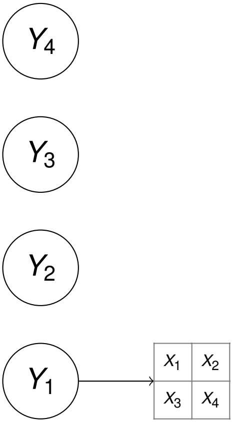
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

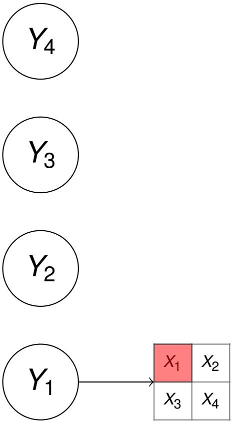
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

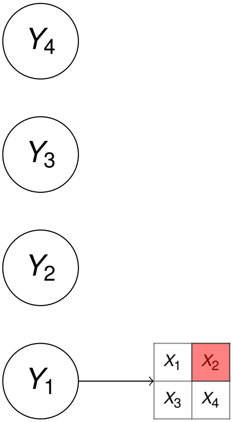
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

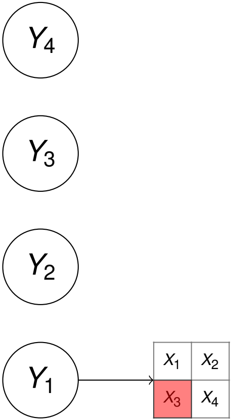
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

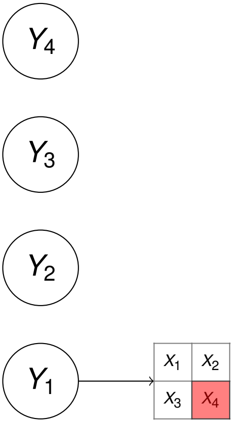
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

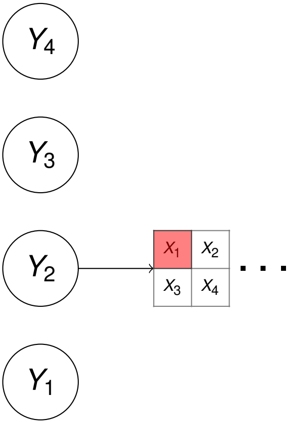
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

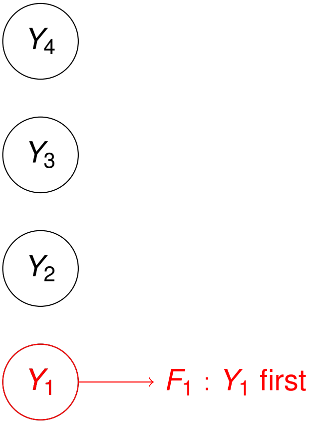
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

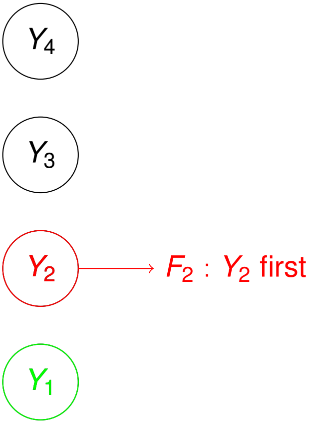
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

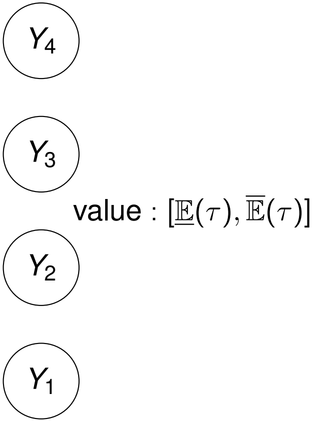
- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



A possible solution

For a trajectory

- For each position Y_i, compute probability bounds that at least one of its cell is occupied
- Along a trajectory, compute successive probability bound that *ith* position is the first occupied (if occupied, no need to look beyond)
- Compute utility of a trajectory from this information (each actions=same utilities but different states)



Computations for one state

- $A =$ at least one cell among n occupied

$$
\begin{aligned}
A &= x_1 \vee x_2 \vee \ldots \vee x_n \\
&= x_1 \vee (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3) \ldots \vee (\wedge_{i=1}^{n-1} \neg x_i \bigwedge x_n) \\
&= A_1 \vee A_2 \vee \ldots \vee A_n \text{ with } A_i \cap A_j = \emptyset
\end{aligned}
$$

- A is monotonic (if one cell become occupied, A cannot become false)
- Since A_i are disjoint

$$
\underline{P}(A) = \sum_{i=1}^{n} \left(Bel(x_i) \prod_{j=1}^{i-1} Pl(\neg x_j) \right).
$$

- For each position i, we get $\underline{P}(y_i), \overline{P}(y_i)$ of being "occupied"

Numerical example

	X_1	X_2	X_3	X_4
$Bel(x_i)$	0.2	0.1	0	0.6
$Pl(x_i)$	0.2	1	0.1	0.7

$$\begin{aligned} Bel(y) =& Bel(x_1) + Bel(x_2)Pl(\neg x_1) + Bel(x_3)Pl(\neg x_1)Pl(\neg x_2) \\ &+ Bel(x_4)Pl(\neg x_1)Pl(\neg x_2) + Pl(\neg x_3). \\ =& 0.2 + 0.1(1-0.2) + 0(1-0.2)(1-0.1) \\ &+ 0.6(1-0.2)(1-0.1)(1-0) \\ =& 0.712 \end{aligned}$$

Trajectory computations

- $F_i = ith$ position is first occupied

$$F_i = \wedge_{j=1}^{i-1} \neg y_j \wedge y_i$$

- F_{k+1} = totally free trajectory

$$F_{k+1} = \wedge_{j=1}^{k} \neg y_j$$

- F_i conjunction of disjoint events $\implies$

$$\underline{P}(F_i) = Bel(y_i) \prod_{j=1}^{i-1} Bel(\neg y_j)$$

  and the same for $\overline{P}(F_i)$
- If k positions, $F_1, \ldots, F_{k+1}$ form a partition of possible states $\implies$ $\underline{P}(F_i), \overline{P}(F_i)$ are probability intervals on atoms F_i (may not be a belief function! $\rightarrow$ see Didier talk and exercices)

Numerical example

Four possible positions, third likely to be occupied

	Y_1	Y_2	Y_3	Y_4
$\underline{P}(y_i)$	0.1	0.3	0.712	0
$\overline{P}(y_i)$	0.2	0.5	1	1

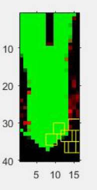
from which result the bounds

	F_1	F_2	F_3	F_4	F_5
$\underline{P}(F_i)$	0.1	0.24	0.28	0	0
$\overline{P}(F_i)$	0.2	0.45	0.63	0.18	0.18

First occupied positions for this trajectory rather Y_2 or Y_3

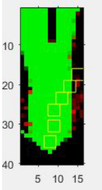
Example of final result

$$u = [-1,\ -0.5,\ -0.25,\ 0,\ 0.25,\ 0.5,\ 1,\ 2,\ 4,\ 8,\ 16]$$

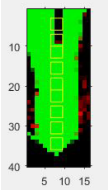


Traj. 1

...

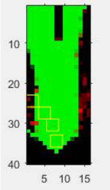


Traj. 4

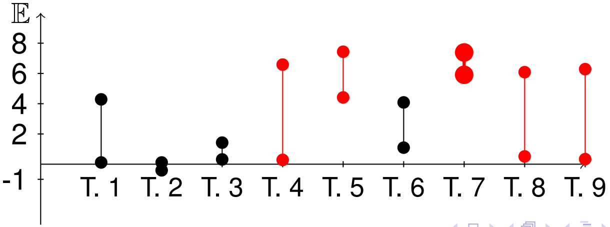


Traj. 6

...



Traj. 9



Some final words

- This talk concerned very basic combinations of logic, Boolean formulas and belief functions, with a focus on (some) computational aspects
- Much more to say, both on interpretations (ask Didier, Davide, Barbara) and on computational aspects (on compilation, SAT, . . .)
- More importantly, much more to do, both
  - theoretically, as many interplays between logic and belief functions remain to be explored
  - practically, as many applications (not only those using logic explicitly) can be seen as inferences/decisions over Boolean variables

References I

[1] Mohua Banerjee and Didier Dubois.
A simple logic for reasoning about incomplete knowledge.
International Journal of Approximate Reasoning, 55(2):639–653, 2014.

[2] Randal E Bryant.
Symbolic boolean manipulation with ordered binary-decision diagrams.
ACM Computing Surveys (CSUR), 24(3):293–318, 1992.

[3] Randal E Bryant.
Binary decision diagrams.
In *Handbook of model checking*, pages 191–217. Springer, 2018.

[4] Laurence Cholvy.
Towards a logical belief function theory.
Journal of Applied Logic, 13(4):441–457, 2015.

[5] Inés Couso, Serafín Moral, and Peter Walley.
A survey of concepts of independence for imprecise probabilities.
Risk, Decision and Policy, 5(2):165–181, 2000.

References II

[6] Fabio Cozman.
The joy of probabilistic answer set programming.
In *International Symposium on Imprecise Probabilities: Theories and Applications*, pages 91–101, 2019.

[7] Florence Dupin De Saint-Cyr, Jérôme Lang, and Thomas Schiex.
Penalty logic and its link with dempster-shafer theory.
In *Uncertainty Proceedings 1994*, pages 204–211. Elsevier, 1994.

[8] Sébastien Destercke and Didier Dubois.
Idempotent conjunctive combination of belief functions: Extending the minimum rule of possibility theory.
Information Sciences, 181(18):3925–3945, 2011.

[9] Didier Dubois and Henri Prade.
Possibilistic logic: a retrospective and prospective view.
Fuzzy sets and Systems, 144(1):3–23, 2004.

[10] Anthony Hunter and Sébastien Konieczny.
Approaches to measuring inconsistent information.
In *Inconsistency tolerance*, pages 191–236. Springer, 2005.