Forest Orders

DA2PL 2026

Bruxelles, April 17th 2026

Jérome Gaigne, Khaled Belahcène, Sylvain Lagrue

Modeling Ignorance – What?

Comparative statements

- Preference between alternatives
- Plausibility of Beliefs

Modeling Ignorance – Why?

Dilemma : *do you prefer your mum or your dad?*

also Sophie's choice, Moral Machine, absurdtrolleyproblems,...

Imprecise Knowledge : *how much sand is a heap?*

related to Intransitivity of Preference : *How much sugar is too much?*

Trustworthy AI abstains from deciding

systems that know they don't know and cannot infer from what they don't know

Provably Beneficial AI [Russel] defers to users

Artificial agents that align themselves to human goals by design

Modeling Ignorance – How?

Using an outranking relation $\precsim$ reflexive and transitive (partial order)

$x \precsim y$ and not $y \precsim x$: strict preference of x over y

$y \precsim y$ and not $x \precsim y$: strict preference of y over x

$x \precsim y$ and $y \precsim x$: indifference between x and y

neither $x \precsim y$ nor $y \precsim x$: incomparability between x and y

Expressive but cumbersome / intractable

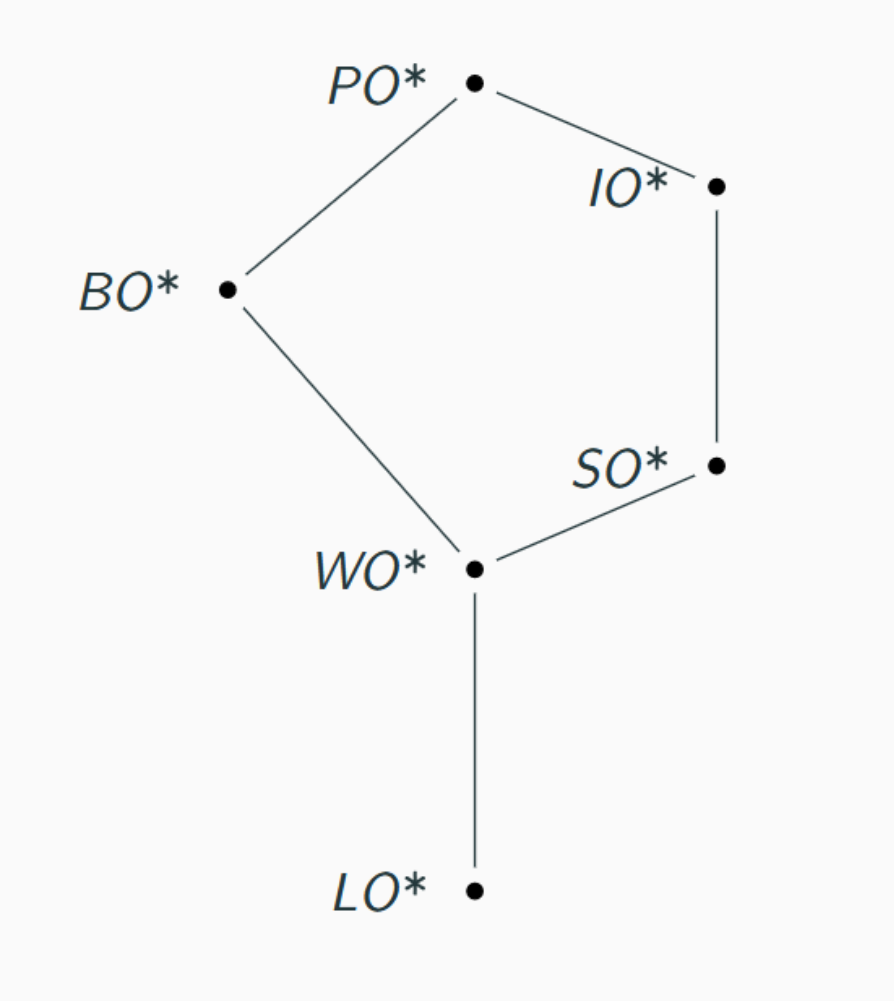
Modeling Ignorance – How?

Using a numeric representation

Baseline : Scores natively represent **Total Preorders / Weak Orders**

- Built-in representation of **transitivity** and **totality**
- Ignorance = indistinguishability inside each bucket

Moving along the expressiveness ladder



- Moving down
  - **Linear orders** : no ties, no ignorance
- Moving up by enriching the representation
  - **SemiOrders**

    Alternatives represented by fixed-size intervals

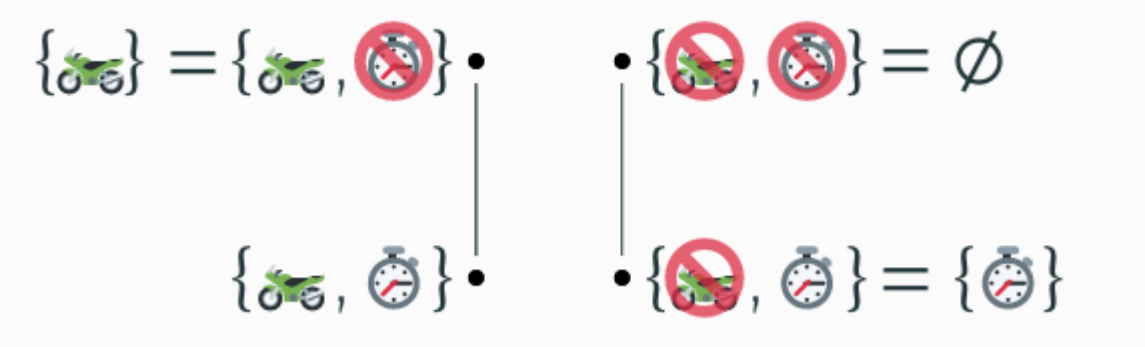
    Cannot conclude when intervals intersect
  - **Interval Orders**

    Intervals can have any size

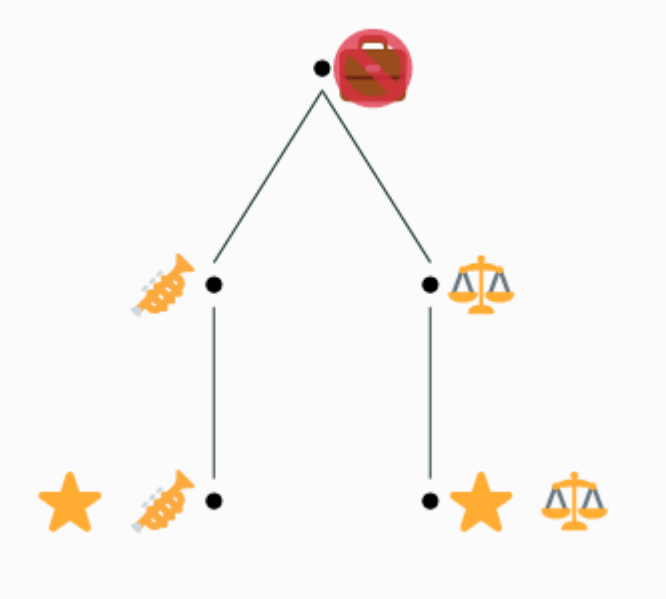
This branch models **Ignorance as Imprecision**

No 2+2 ? How sad!

- **Beliefs :** Bob the Biker

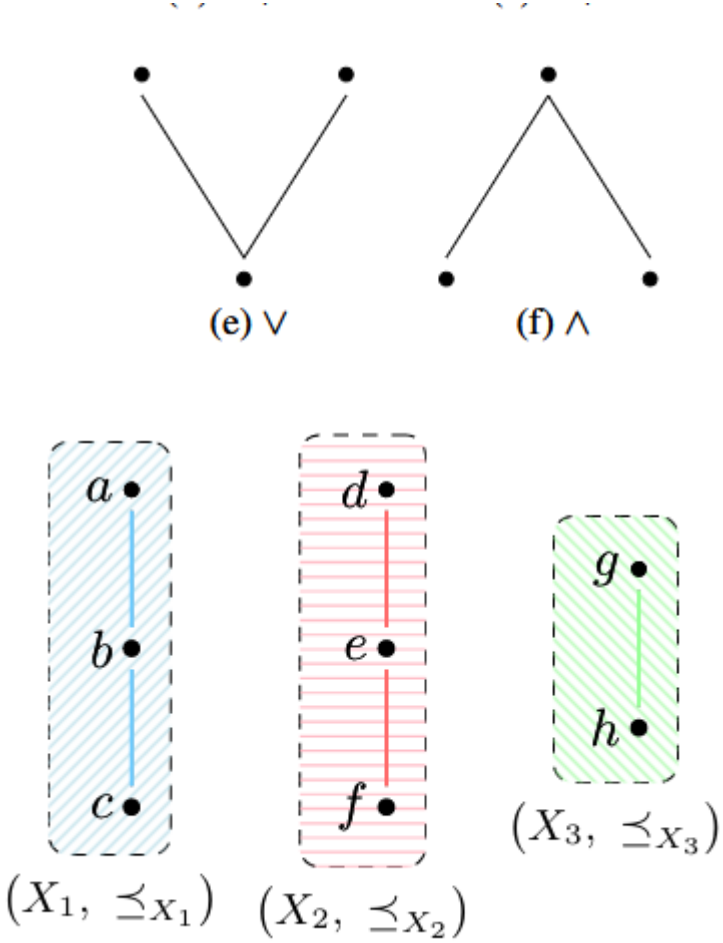


- **Preferences :** Raz'career choice, extended



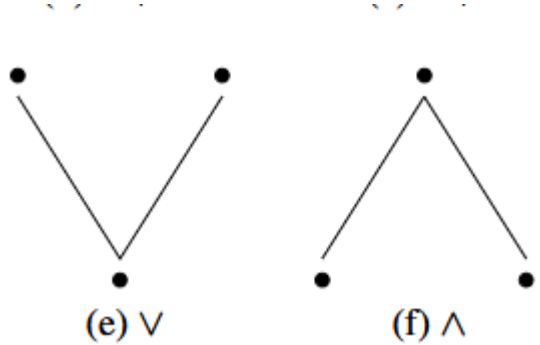
Partitioned linear orders [Gaigne24]

- Two forbidden patterns
- Equivalent to impose *transitivity of comparability*
- Leads to a partition into comparability classes that are linearly ordered
- Has linear dimension of at most 2
- Can be captured inside a principled belief revision framework



Forest Orders

One forbidden pattern amongst



Theorem. For a poset P= (X,⪯), the following conditions are equivalent:

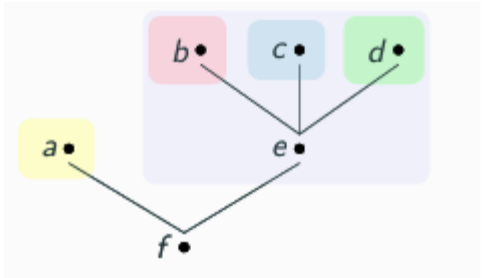
1. Free of ∧–shape
2. Predecessor injective
3. Left comparable (downward total)

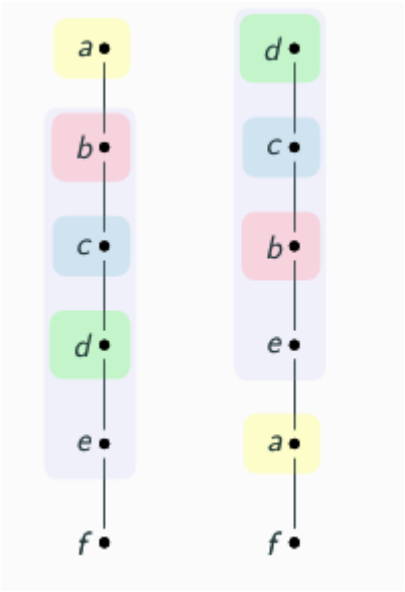
If well-founded, disjoint union of out-tree ordered sets is also equivalent to 1+2+3

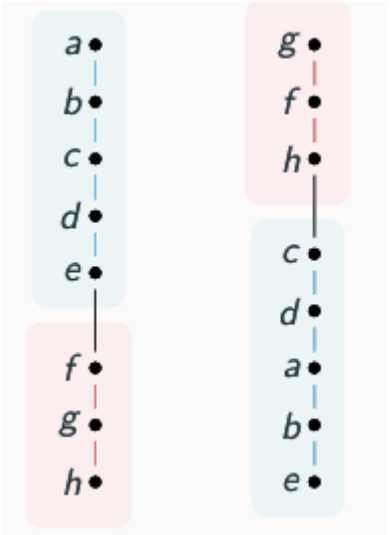
Definition. A poset P that satisfies the conditions above is an **out-forest order**.

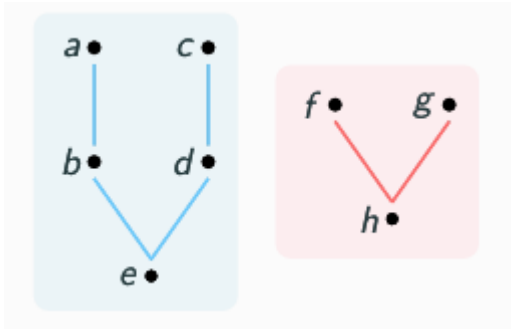
Forest Orders are Bilinear Orders: Proof

For a single tree :
perform 2 DFS traversing the children in reverse order









For a forest :
decorate with a dummy root

Forest Orders are Bilinear Orders: Consequences

Representation :

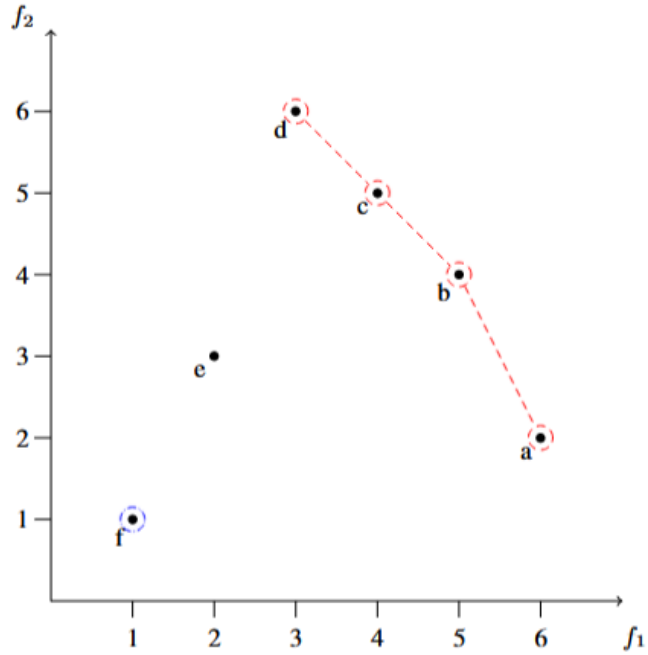
- any FO can be represented with two score functions
- Preference occurs iff the two scores agree

Optimization :

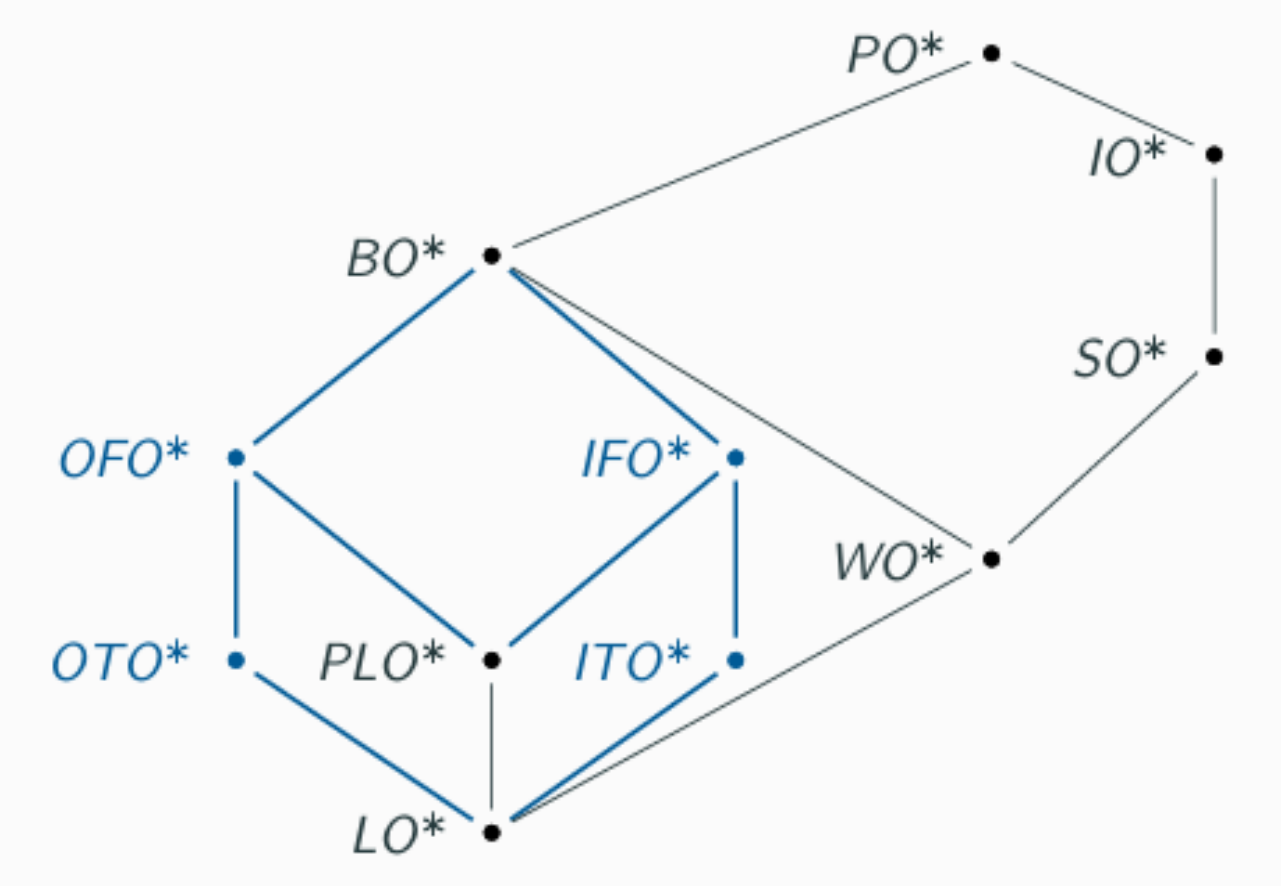
- the minimal elements of an out-forest order form a bi-objective Pareto front
- an additional peculiarity makes them slightly easier to compute

Axiomatization :

- The forbidden pattern is ternary
- as with PLOs, the forbidden pattern can be captured in a principled Belief Revision framework
- BO, in general, cannot be described by a finite collection of forbidden patterns/logical axioms



The landscape of not-so-partial orders



- Right branch:
  Ignorance as Imprecision
- Left branch:
  Ignorance as Conflict

Usual Bi-objective Optimization

Algorithm 1: Finding the Pareto front in a two criteria decision problem

Data: A set X and two score functions f_1, f_2

Result: A list L of the Pareto efficient elements of X w.r.t. f_1 and f_2

```
1 SortInPlace(X);
2 L ← NewStack();
3 L.push(min_{x∈X} f1(x));
4 foreach x ∈ X \ {L.top()} do
5 |   if f2(x) ≤ f2(L.top()) then
6 |   |   L.push(x);
7 |   end
8 end
```

Out-Forest Order Optimization

Algorithm 2: Finding the maximal elements of $P = (X, \preceq_P)$, an OFOset

Data: A set X and two score functions f_1, f_2

Result: A list L of the Pareto efficient elements of X w.r.t. f_1 and f_2

```
1 SortInPlace(X);
2 L ←NewList();
3 L.add(max_{x∈X} f1(x));
4 W ←sliding_window(2, X);
5 foreach (a, b) ∈ W do
6 |   if f2(b) ≤ f2(a) then
7 |   |   L.add(a);
8 |   end
9 end
```