

ABA LEARNING

with APPLICATIONS
to NEURO-SYMBOLIC AI

Emanuele De Angelis, Cnr-IASI, Italy

Maurizio Proietti, Cnr-IASI, Italy

Francesca Toni, IMPERIAL, UK

Meeting Progetto GNCS
Rimini, 5-6 February 2026

ABA LEARNING **with APPLICATIONS** **to NEURO-SYMBOLIC AI**

Emanuele De Angelis, Cnr-IASI, Italy

Maurizio Proietti, Cnr-IASI, Italy

Francesca Toni, IMPERIAL, UK

Meeting Progetto GNCS
Rimini, 5-6 February 2026

Logic-based learning
of ABA frameworks from
background knowledge
+
positive & negative
examples

Algorithm based on
transformation rules,
implemented in
Answer Set Programming

ABA LEARNING **with APPLICATIONS** **to NEURO-SYMBOLIC AI**

Emanuele De Angelis, Cnr-IASI, Italy

Maurizio Proietti, Cnr-IASI, Italy

Francesca Toni, IMPERIAL, UK

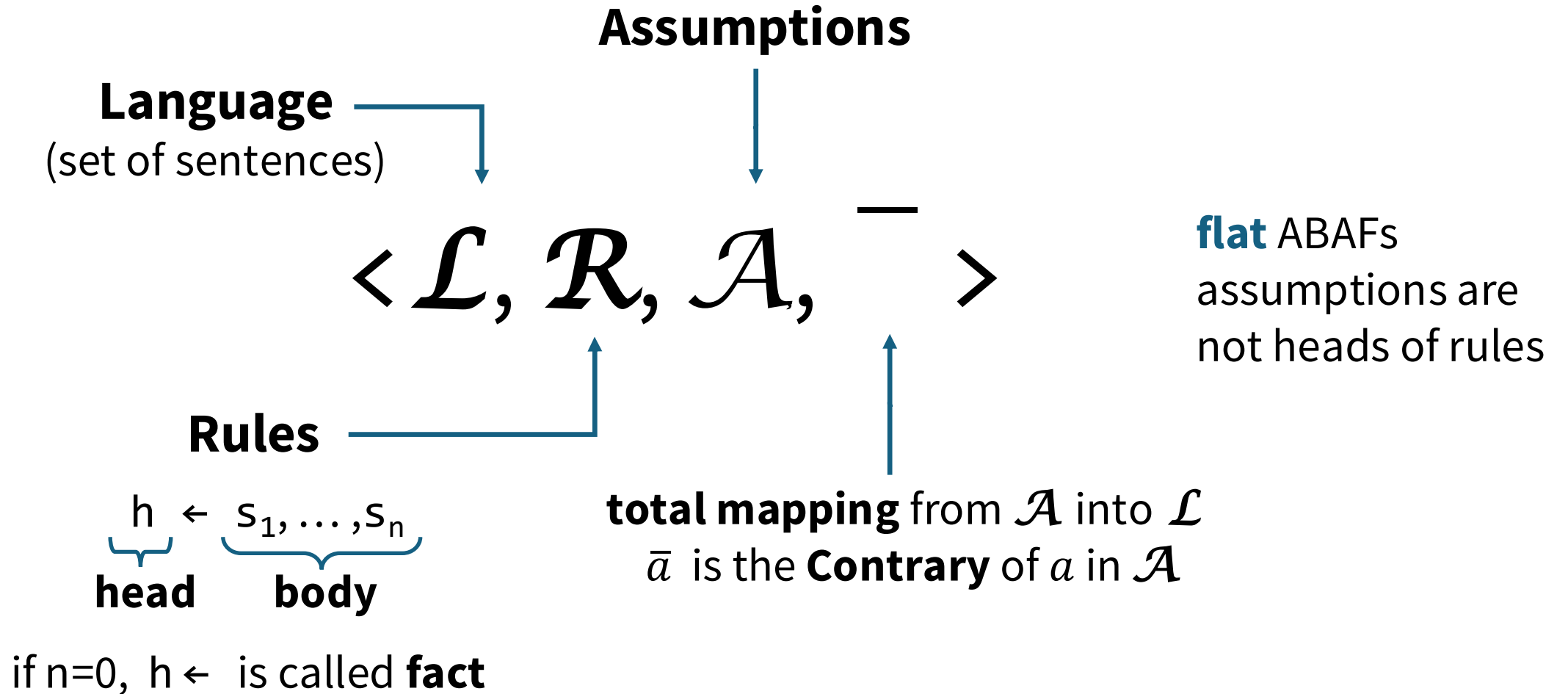
Meeting Progetto GNCS
Rimini, 5-6 February 2026

**ARgumentation-based
Explainable recommender
System (ARES)**

**Object-Centric
Neuro-Argumentative
Learning (OC-NAL)**

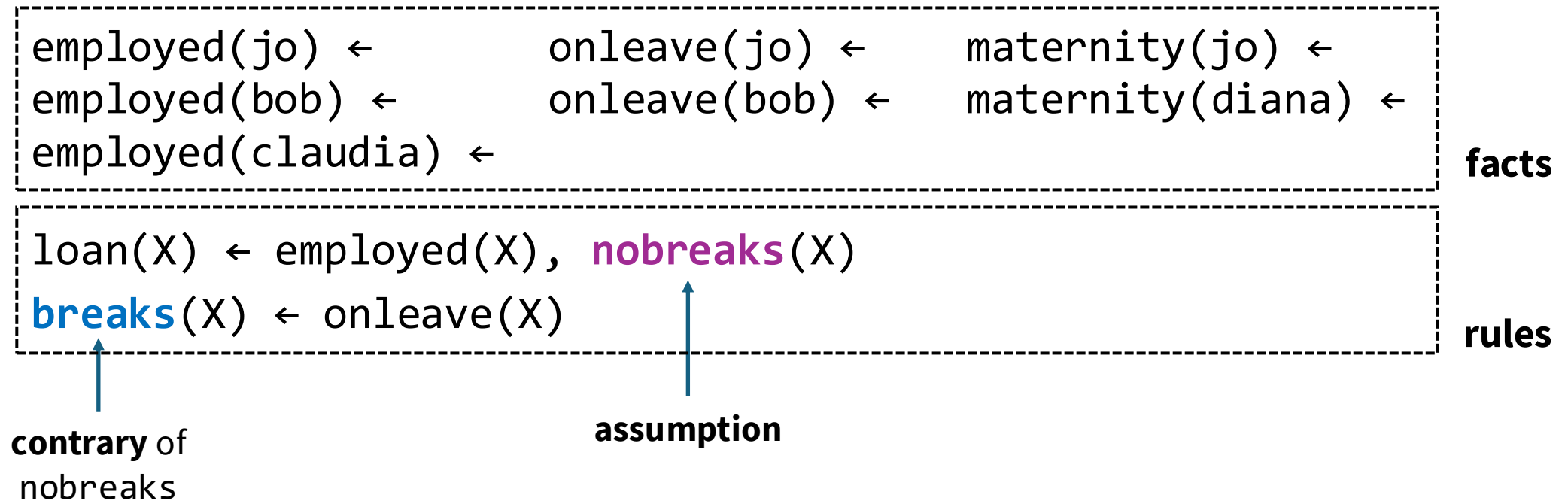
**Neuro-Argumentative
Learning with Legal Text**

ABA FRAMEWORKS



ABA FRAMEWORKS

an example ...



nobreaks(X) renders the rule defeasible:
it can be applied only if **breaks**(X) cannot be derived

ABA FRAMEWORKS - SEMANTICS

“**acceptable**” extensions:
sets of arguments able to
 “defend” themselves from “attacks”
 (as determined by the chosen semantics)

- **Arguments** are **deductions** of claims using **rules** and supported by **assumptions**
- **Attacks** are directed at the assumptions in the support of arguments

```

employed(jo) ←      onleave(jo) ←      maternity(jo) ←
employed(bob) ←     onleave(bob) ←     maternity(diana) ←
employed(claudia) ←
loan(X) ← employed(X), nobreaks(X)
breaks(X) ← onleave(X)
  
```

```

{
  attacks {
    arg1: { nobreaks(jo) } ⊢ loan(jo)
    arg2: { nobreaks(bob) } ⊢ loan(bob)
    arg3: { nobreaks(claudia) } ⊢ loan(claudia)
    arg4: { } ⊢ breaks(jo)
    arg5: { } ⊢ breaks(bob)
  }
}
  
```

We focus on **stable extensions**

any set of arguments S that

1. do not attack each other (conflict-free)
2. S attacks all arguments it does not contain

Accepted claims: loan(claudia)

Rejected claims: loan(jo)
 loan(bob)

BRAVE ABA LEARNING PROBLEM

Given

1. ABA framework $\mathbf{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\quad} \rangle$ (**background knowledge**)
with at least one stable extension
2. $\mathbf{E}_p = \{ \text{positive examples} \}$
3. $\mathbf{E}_n = \{ \text{negative examples} \}$
4. $\mathbf{T} = \{ \text{learnable predicates} \}$

find $\mathbf{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}', \bar{\quad} \rangle$ with **a stable extension** S such that

- i. $\mathbf{F} \subseteq \mathbf{F}'$
- ii. **positive** are **covered**: every positive has an argument in S
- iii. **negative** are **not covered**: no negative has an argument in S

\mathbf{F}' is a **solution** to the brave ABA learning problem

CAUTIOUS ABA LEARNING PROBLEM

Given

1. ABA framework $\mathbf{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\quad} \rangle$ (**background knowledge**)
with at least one stable extension
2. $\mathbf{E}_p = \{ \text{positive examples} \}$
3. $\mathbf{E}_n = \{ \text{negative examples} \}$
4. $\mathbf{T} = \{ \text{learnable predicates} \}$

find $\mathbf{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}', \bar{\quad} \rangle$ with **at least one stable extension**

- i. $\mathbf{F} \subseteq \mathbf{F}'$
- ii. **positive** are **covered**: every positive has an argument in \mathbf{C}
- iii. **negative** are **not covered**: no negative has an argument in \mathbf{C}

$$\mathbf{C} = \bigcap_k^n S_i$$

S_1, \dots, S_n : stable extensions of \mathbf{F}'

\mathbf{F}' is a **solution** to the **cautious** ABA learning problem

ABA LEARNING VIA TRANSFORMATION RULES

Learning ABA frameworks relies upon a set of **transformation rules**

$$\langle \mathcal{L}_1, \mathcal{R}_1, \mathcal{A}_1, \overset{-1}{\phantom{\mathcal{A}_1}} \rangle \longrightarrow \langle \mathcal{L}_2, \mathcal{R}_2, \mathcal{A}_2, \overset{-2}{\phantom{\mathcal{A}_2}} \rangle \longrightarrow \dots \longrightarrow \langle \mathcal{L}_n, \mathcal{R}_n, \mathcal{A}_n, \overset{-n}{\phantom{\mathcal{A}_n}} \rangle$$

background knowledge intensional solution

$\longrightarrow \in \{ \text{Rote Learning, Folding, Assumption Introduction, Subsumption} \}$

learnt rules **do not**
make explicit
reference to specific
values in the universe

A **strategy** controls the order of application of the transformation rules

ABA LEARNING at work

Language
{ employed(X)←, onleave(X)←,
maternity(X)←, loan(X)← }

Assumptions
{ nobreaks(X) }

Contraries
{ nobreaks(X) = breaks(X) }

1. Background Knowledge $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{\quad} \rangle$

Rules
{ employed(jo)←, employed(bob)←, employed(claudia)←,
onleave(jo)←, onleave(bob)←,
maternity(jo)←, maternity(diana)← }

$X \in \{claudia, jo, bob, diana\}$

2. Positive examples $\mathbf{E_p} = \{ \text{loan}(claudia) \}$

3. Negative examples $\mathbf{E_n} = \{ \text{loan}(bob) \}$

4. Learnable Predicates $\mathbf{T} = \{ \text{loan} \}$

TRANSFORMATION RULES at work

ROTE LEARNING

Add **facts**

- from **positive** examples
 - for **contraries** of **assumptions**
- to get a (**non-intensional**) **solution**

It's enough to learn

$\text{loan}(X) \leftarrow X=\text{claudia}$

$\mathbf{E_p} = \{ \text{loan}(\text{claudia}) \}$

to get

$\mathcal{R}' = \mathcal{R} \cup \{ \text{loan}(X) \leftarrow X=\text{claudia} \}$

TRANSFORMATION RULES at work

FOLDING

Towards an **intensional** solution ...

Generalise

`loan(X) ← X=claudia`

to

`loan(X) ← employed(X)`

by using

`employed(X) ← X=claudia`

WARNING

It also constructs an argument for a **negative** example: `loan(bob)`

ABA LEARNING is **parametric** w.r.t. the folding strategy

It includes a portfolio of strategies, such as “nondeterministic” and “greedy” folding

TRANSFORMATION RULES at work

ASSUMPTION INTRODUCTION

Repairing the ABA framework to get a solution ...

Add an **assumption** to avoid

- **rejecting** a **positive** example
- **accepting** a **negative** example

$\text{loan}(X) \leftarrow \text{employed}(X), \text{nobreaks}(X)$

with contrary
breaks(X)

AND REPEAT!

Rote Learning

`breaks(X) ← X=bob`

Folding

`breaks(X) ← onleave(X)`

**No more rules to learn:
LEARNING COMPLETED!**

`employed(jo) ←
employed(bob) ←
employed(claudia) ←`

`onleave(jo) ←
onleave(bob) ←`

`maternity(jo) ←
maternity(diana) ←`

Rules in the Background Knowledge

`loan(X) ← employed(X), nobreaks(X)
breaks(X) ← onleave(X)`

Learnt rules

Algorithm 1: RASP-ABALearn

```

Input:  $\langle \mathcal{R}_0, \mathcal{A}_0, \bar{\ }^0, \mathcal{E}^+, \mathcal{E}^-, \mathcal{E}_C^+, \mathcal{E}_C^-, \mathcal{T} \rangle$ : redress problem
Output:  $\langle \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ : incremental redress relative to  $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$ 
1  $\mathcal{R} := \mathcal{R}_0$ ;  $\mathcal{A} := \mathcal{A}_0$ ;  $\bar{\ } := \bar{\ }^0$ ;  $\mathcal{R}_l := \emptyset$ ;
2 RoLe(); Gen(); return  $\langle \mathcal{R}, \mathcal{A}, \bar{\ } \rangle$ ;
3 Procedure RoLe()
4  $P := ASP(\langle \mathcal{R}, \mathcal{A}, \bar{\ } \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \mathcal{T})$ ;
5 if  $\neg sat(P)$  then
6   fail;
7 else
8    $S := as(P)$ ;
9   // R1. Rote Learning
10  foreach  $newp(t) \in S$  do
11     $\mathcal{R}_l := \mathcal{R}_l \cup \{p(X) \leftarrow X=t\}$ ;
12  end
13 Procedure Gen()
14 foreach  $\rho : (p(X) \leftarrow X=t) \in \mathcal{R}_l$  do
15    $\mathcal{R}_l := \mathcal{R}_l \setminus \{\rho\}$ ;
16   // R4. Fact Subsumption
17   if  $\neg sat(ASP(\langle \mathcal{R} \cup \mathcal{R}_l, \mathcal{A}, \bar{\ } \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$  then
18     // R2 w/ R3. Folding with Assumption Introduction
19      $\langle \rho_g, \alpha(X), C_\alpha \rangle := FoldingWasmIntro(\rho)$ ;
20      $\mathcal{R} := \mathcal{R} \cup \{\rho_g\}$ ;
21      $\mathcal{A} := \mathcal{A} \cup \{\alpha(X)\}$ ;
22      $\overline{\alpha(X)} := c\_ \alpha(X)$ ;
23     // R1. Rote Learning
24     foreach  $c\_ \alpha(t) \in C_\alpha$  do
25        $\mathcal{R}_l := \mathcal{R}_l \cup \{c\_ \alpha(X) \leftarrow X=t\}$ ;
26     end
27   end
28 end
29 Function FoldingWasmIntro( $\rho$ )
30 // R2. Folding
31 while foldable( $\rho, \mathcal{R}$ ) do
32    $\rho := fold(\rho, \mathcal{R})$ ;
33 end
34 // R3. Assumption Introduction
35 Let  $\rho$  be  $H \leftarrow B$ ;  $X := vars(B)$ ;
36 if there exists  $\alpha(X) \in \mathcal{A}$  relative to  $B$  then
37    $\rho_g := H \leftarrow B, \alpha(X)$ ;  $C_\alpha := \emptyset$ ;
38   if  $\neg sat(ASP(\langle \mathcal{R} \cup \{\rho\}, \mathcal{A}, \bar{\ } \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$  then
39     fail;
40   end
41 else // introduce an assumption  $\alpha(X)$ , with a new predicate  $\alpha$ 
42    $\rho_g := H \leftarrow B, \alpha(X)$ ;
43    $F := \langle \mathcal{R} \cup \{\rho\}, \mathcal{A} \cup \{\alpha(X)\}, \bar{\ } \cup \{\alpha(X) \rightarrow c\_ \alpha(X)\} \rangle$ ;
44    $C_\alpha := \{c\_ \alpha(X) \mid c\_ \alpha(X) \in as(ASP(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \{c\_ \alpha\}))\}$ ;
45 end
46 return  $\langle \rho_g, \alpha(X), C_\alpha \rangle$ ;

```

A GLIMPSE OF IMPLEMENTATION via ASP

- **ASP encoding**

```

loan(X) :- employed(X), nobreaks(X). ...
nobreaks(X) :- employed(X), not breaks(X).

```

```

{ breaksP(X) } :- employed(X).
breaks(X) :- breaksP(X).
#minimize{1,X: breaksP(X)}.
:- not loan(claudia).
:- loan(bob).

```

learning facts
for contraries

- **Answer sets**

(1-to-1 correspondence with **Stable extensions**)

```
{ breaksP(bob), ... }, ...
```

- **Rote learning**

```
breaks(X) ← X=bob
```



SWI Prolog

+

Clingo (ASP)



https://github.com/ABALearn/aba_asp

ARgumentation-based Explainable recommender System (ARES)

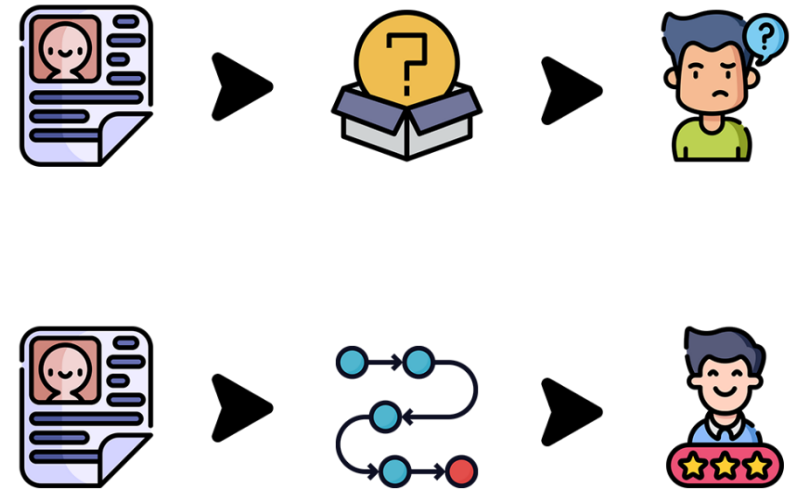
Joint work with

Riccardo Felici, Alessio Ferrato, Giuseppe Sansonetti

Università Roma Tre

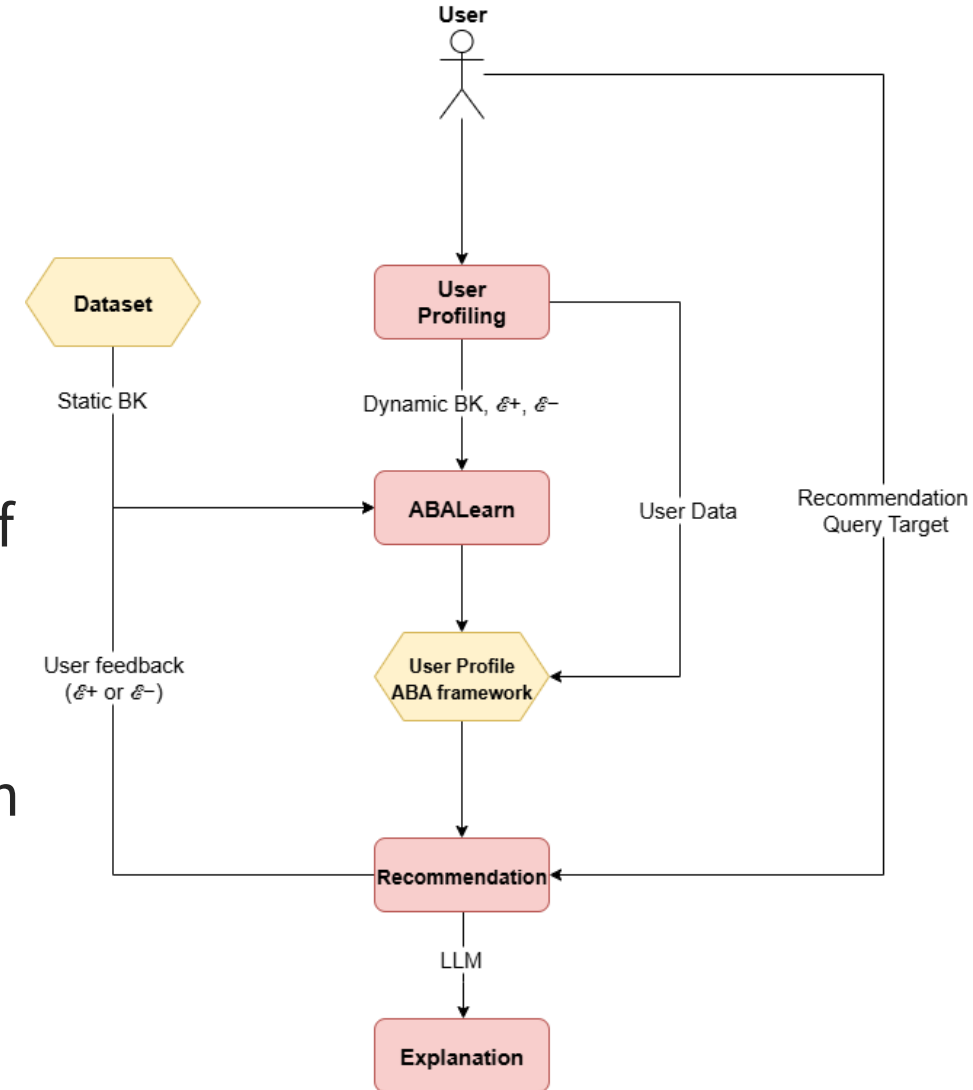
MOTIVATION

- Recommender Systems guide us through vast amounts of information, but they are often **black boxes**
- Critical issues:
 - **Limited user trust**: Why does the system recommends this item?
 - **Poor user experience**: Why is the system giving a recommendation that I dislike?
- **ARES**: A recommender system that provides **transparent** and **explainable** recommendations based on Assumption-Based Argumentation



ARES ARCHITECTURE

1. **User Profiling** module: collects user preferences (likes/dislikes) and feedback
2. **ABALearn** module: **learns an ABAF** that represents the user's profile
3. **Recommender** module: generates a ranked list of recommendations by using the learnt ABAF
4. **Explanation** module: uses **Argument Trees** and **LLMs** to translate the **formal ABAF** representation into a **human-readable explanation**
5. **Chatbot**: manages the interaction between the user and the system in NL



FROM USER PROFILES TO ABA FRAMEWORKS

BACKGROUND KNOWLEDGE & EXAMPLES

From the **dataset** of perfumes

oriental(A) ← A=vanilla

sweet(A) ← A=sugar

ingredient_of(A, B) ← A=meltine, B=vanilla

ingredient_of(A, B) ← A=velvette, B=sugar

From the **user profile**

like(A) ← ingredient_of(A,B), sweet(B), alpha1(A,B)

sweet(A) ← A=candy

ingredient_of(A, B) ← A=p1, B=candy

Positive examples **E_p = { like(meltine), like(p1) }**

Negative examples **E_n = { like(velvette) }**

- From BK & examples:

like(A) ← ingredient_of(A,B), sweet(B), alpha1(A,B)

c_alpha1(A,B) ← ingredient_of(A,B), B=sugar

like(A) ← ingredient_of(A,B), oriental(B)

- The learnt ABAF accepts all positive examples and no negative example

EXPLANATIONS

Rule: `like_rule_2(A) :- alpha_2(A,B), group_tropical(B), ingredient_of(A,B)`

Reasoning Tree Structure:

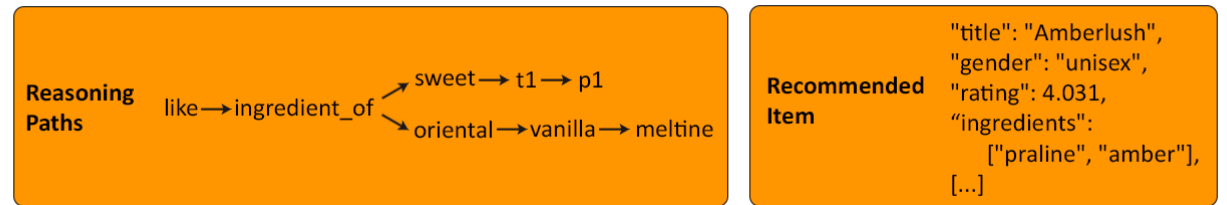
```
graph TD; R["like_rule_2 [2 paths: 1 real, 1 dummy]"] --> I["ingredient_of [2 paths: 1 real, 1 dummy]"]; I --> G["group_tropical [2 paths: 1 real, 1 dummy]"]; G --> IG["ing_grapefruit [1 paths: 1 real, 0 dummy]"]; G --> IT["ing_ttwo [1 paths: 0 real, 1 dummy]"]; IG --> GR["grapefruit [1 paths: 1 real, 0 dummy]"]; GR --> SE["sauvage_elixir (REAL)"]; IT --> TT["ttwo [1 paths: 0 real, 1 dummy]"]; TT --> PT["p_two (DUMMY)"];
```

All Reasoning Paths:

Path 1 REAL
like_rule_2 → ingredient_of → group_tropical → ing_grapefruit → grapefruit → sauvage_elixir

Path 2 DUMMY
like_rule_2 → ingredient_of → group_tropical → ing_ttwo → ttwo → p_two

Tree-like representation of the arguments for the recommended item



Prompt

GOAL: You have to simulate a definition of a chain-of-thoughts from an ABA reasoning tree with the goal of presenting it to the user [...]. From the chain and the details of the perfume, you have to tell why the perfume was recommended to the user. [...]

Return Format: You must return only the chain of thoughts, no header or additional text [...]

WARNINGS: Be careful not to generate information that is not contained within the reasoning paths. Be careful [...].

CONTEXT DUMP: "{reasoning_path}", "{recommended_item}"

Natural Language Explanation

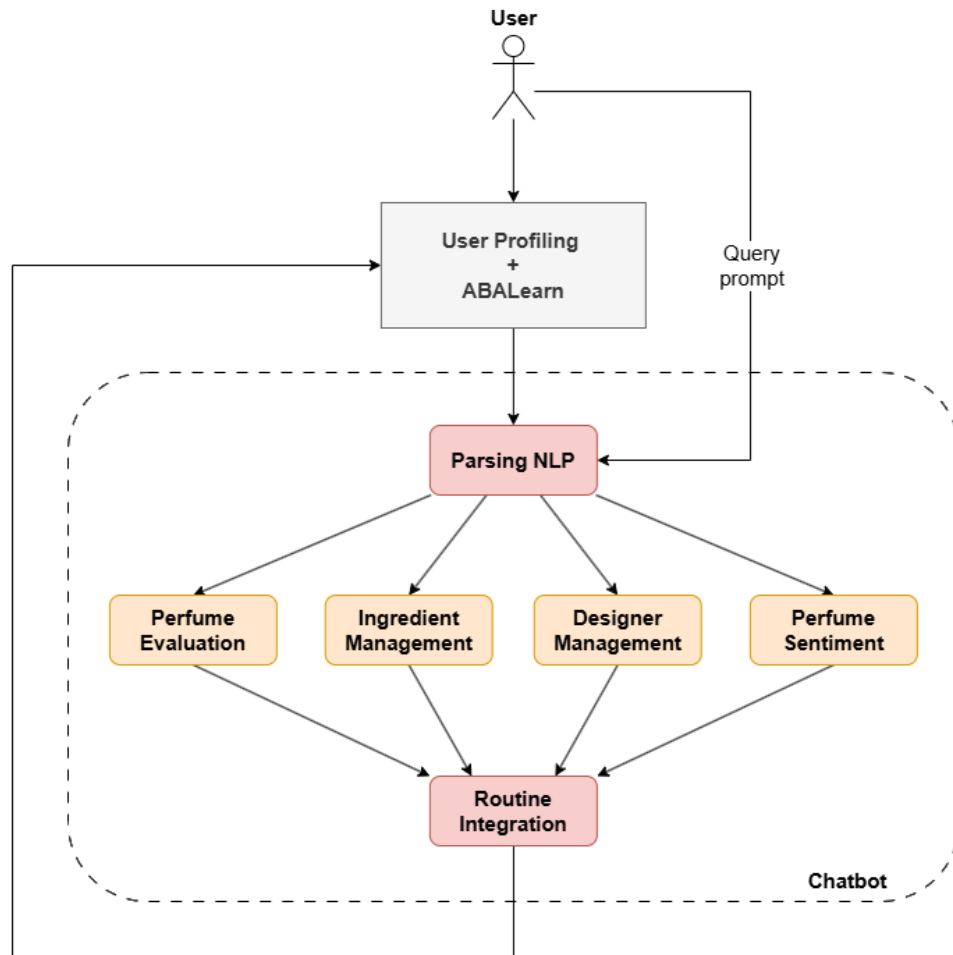
"Given your preferences for sweet notes and your appreciation for the perfume Meltine - which features vanilla and hints at a natural inclination toward oriental notes - Amberlush is a perfect match for you. Its ingredients, praline and amber, reflect your tastes by blending the warmth of sweet accords with the sensual depth of oriental tones. It's a fragrance that's perfect for both day and night, and it shines especially during the warmer seasons."

ITERATIVE LEARNING THROUGH CONTESTATION

ARES implements a dynamic learning process that continuously refines user profiles through **contestability**, the ability for users to **question recommendations** and **explanations**.

1. User receives a recommendation with explanation
2. User can **contest** the recommendation or its explanation
3. ABALearn identifies the rules in the ABAF that lead to the unwanted outcome
4. The system **redresses** the ABAF representing the user profile to resolve conflicts

CHATBOT ARCHITECTURE



I don't like the perfume "Dior Sauvage" that you recommended to me because of the ingredient bergamot

Sauvage added to your disliked perfumes.
Removed Bergamot from your liked ingredients.

Would you recommend Dior Sauvage Elixir to me?

Considering your preferences and the fact that you dislike "Sauvage", it's possible you might not enjoy "Sauvage Elixir" as it's part of the same collection. However, "Sauvage Elixir" leans towards aromatic notes like Lavender and spices like Nutmeg and Cinnamon, which might be different from what you disliked in the original "Sauvage".

This fragrance is suitable for elegant styles and is perceived as more suitable for autumn and winter seasons, leaning towards night wear. Considering the fragrance profile, it could be perceived for an age range from 25 and older.

and would you recommend Zoologist Rabbit?

Type a message...

CONTESTING & REFINING USER PROFILE W/ABALEARN

- User profile:

like(A) ← ingredient_of(A,B), oriental(B)

oriental(B) ← B=amber

But the **amber** ingredient is **not liked** (contestation via chatbot)

- Add a new dummy item **p2** with

ingredient_of(A, B) ← A=p2, B=amber

and **negative example** like(p2)

- ABALearn learns rules:

like(A) ← ingredient_of(A,B), oriental(B), alpha_2(A,B)

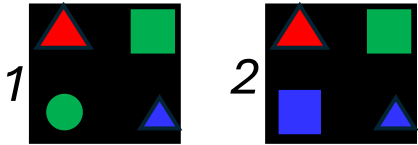
c_alpha_2(A, B) ← ingredient_of(A, B), B=amber

- The learnt ABAF does not accept any like(...) claim for items with **amber**

Object-Centric Neuro-Argumentative Learning (OC-NAL)

ABA Learning for Image Classification

Positive Examples \oplus $c(1), c(2)$



Negative Examples \square $c(3)$



Background Knowledge

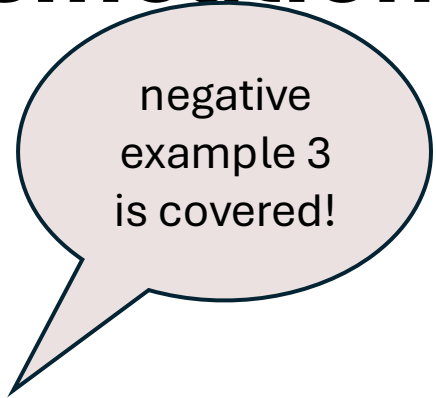
$image(X) \leftarrow X=1$
 $triangle(T) \leftarrow T=t1$ $circle(C) \leftarrow C=c1$
 $red(T) \leftarrow T=t1$ $green(C) \leftarrow C=c1$
 $top_left(T,X) \leftarrow T=t1, X=1$
 $bottom_left(C,X) \leftarrow C=c1, X=1$
 ...
 $image(X) \leftarrow X=3$
 $triangle(T) \leftarrow T=t2$ $triangle(T) \leftarrow T=t3$
 $red(T) \leftarrow T=t2, red(T) \leftarrow T=t3$
 $top_left(T,X) \leftarrow T=t2, X=3$
 $bottom_left(T,X) \leftarrow T=t3, X=3$
 ...
 $on(X,Y,Z) \leftarrow top_left(X,Z), bottom_left(Y,Z)$
 ...

Rote Learning

$c(X) \leftarrow X=1$

Folding

$c(X) \leftarrow image(X), triangle(T), red(T), top_left(T,X)$



Assumption Introduction

$c(X) \leftarrow image(X), triangle(T), red(T), top_left(T,X), alpha(T,X)$

Rote Learning

$alpha_con(T,X) \leftarrow X=3, T=t2$

Folding

$alpha_con(T,X) \leftarrow image(X), triangle(T), red(T), top_left(T,X),$
 $triangle(T'), red(T'), bottom_left(T',X)$

Folding

$alpha_con(T,X) \leftarrow image(X), triangle(T), red(T),$
 $triangle(T'), red(T'),$
 $on(T,T',X)$

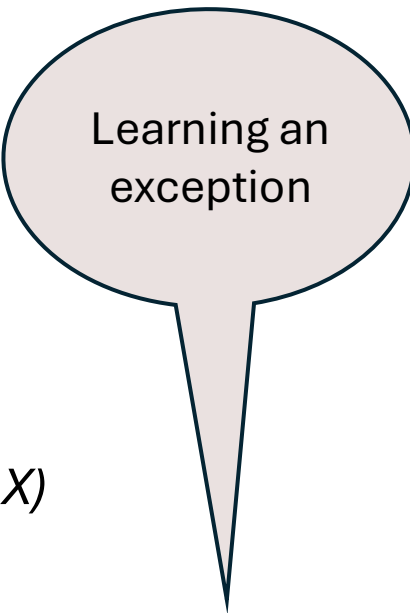
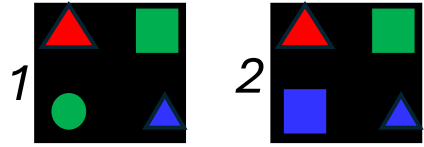
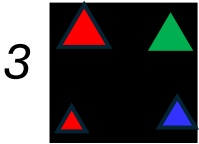


Image classification as argumentative reasoning

Positive Examples \oplus $c(1), c(2)$



Negative Examples \square $c(3)$



Background Knowledge

$image(X) \leftarrow X=1$
 $triangle(T) \leftarrow T=t1$ $circle(C) \leftarrow C=c1$
 $red(T) \leftarrow T=t1$ $green(C) \leftarrow C=c1$
 $top_left(T,X) \leftarrow T=t1, X=1$
 $bottom_left(C,X) \leftarrow C=c1, X=1$
 ...
 $image(X) \leftarrow X=3$
 $triangle(T) \leftarrow T=t2$ $triangle(T) \leftarrow T=t3$
 $red(T) \leftarrow T=t2, red(T) \leftarrow T=t3$
 $top_left(T,X) \leftarrow T=t2, X=3$
 $bottom_left(T,X) \leftarrow T=t3, X=3$
 ...
 $on(X,Y,Z) \leftarrow top_left(X,Z), bottom_left(Y,Z)$
 ...

Learnt ABA framework

$c(X) \leftarrow image(X), triangle(T), red(T), top_left(T,X), alpha(T,X)$
 $alpha_con(T,X) \leftarrow image(X), triangle(T), red(T),$
 $triangle(T'), red(T'),$
 $on(T,T',X)$

(Arguments for) positive examples are accepted

$\{alpha(t1,1)\} \vdash c(1)$

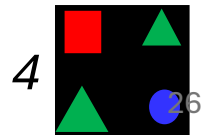
$\{alpha(t1',2)\} \vdash c(2)$

(Arguments for) negative examples are not accepted

$\{ \} \vdash alpha_con(t2,3) \longrightarrow \neg \{alpha(t2,3)\} \vdash c(3)$

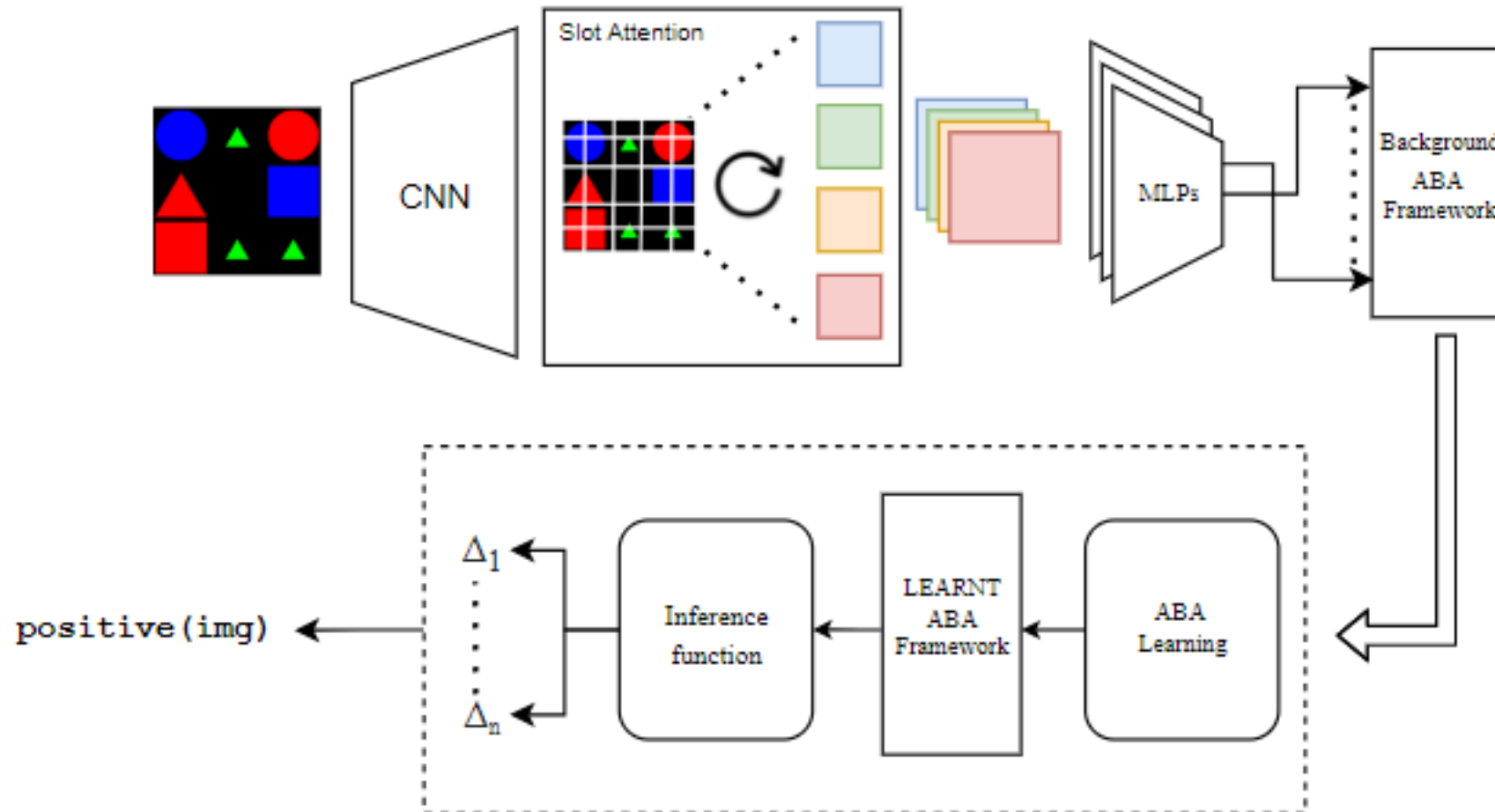
Unknown Example ?

No argument for $c(4)$



Object-Centric Neuro-Argumentative Learning

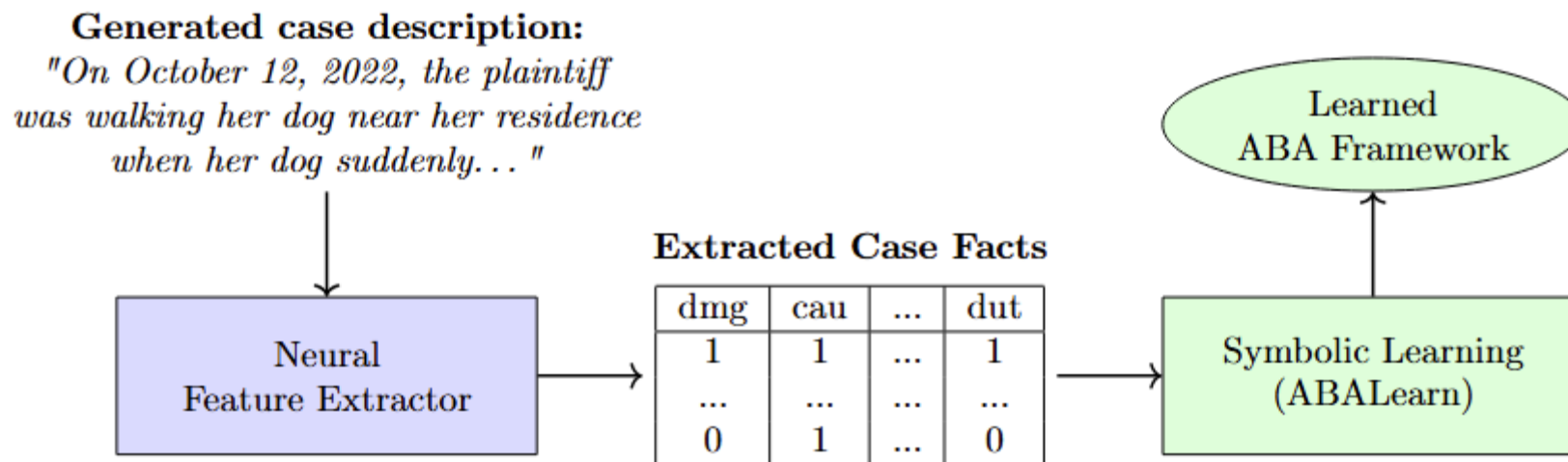
aleatoric uncertainty (noisy data)



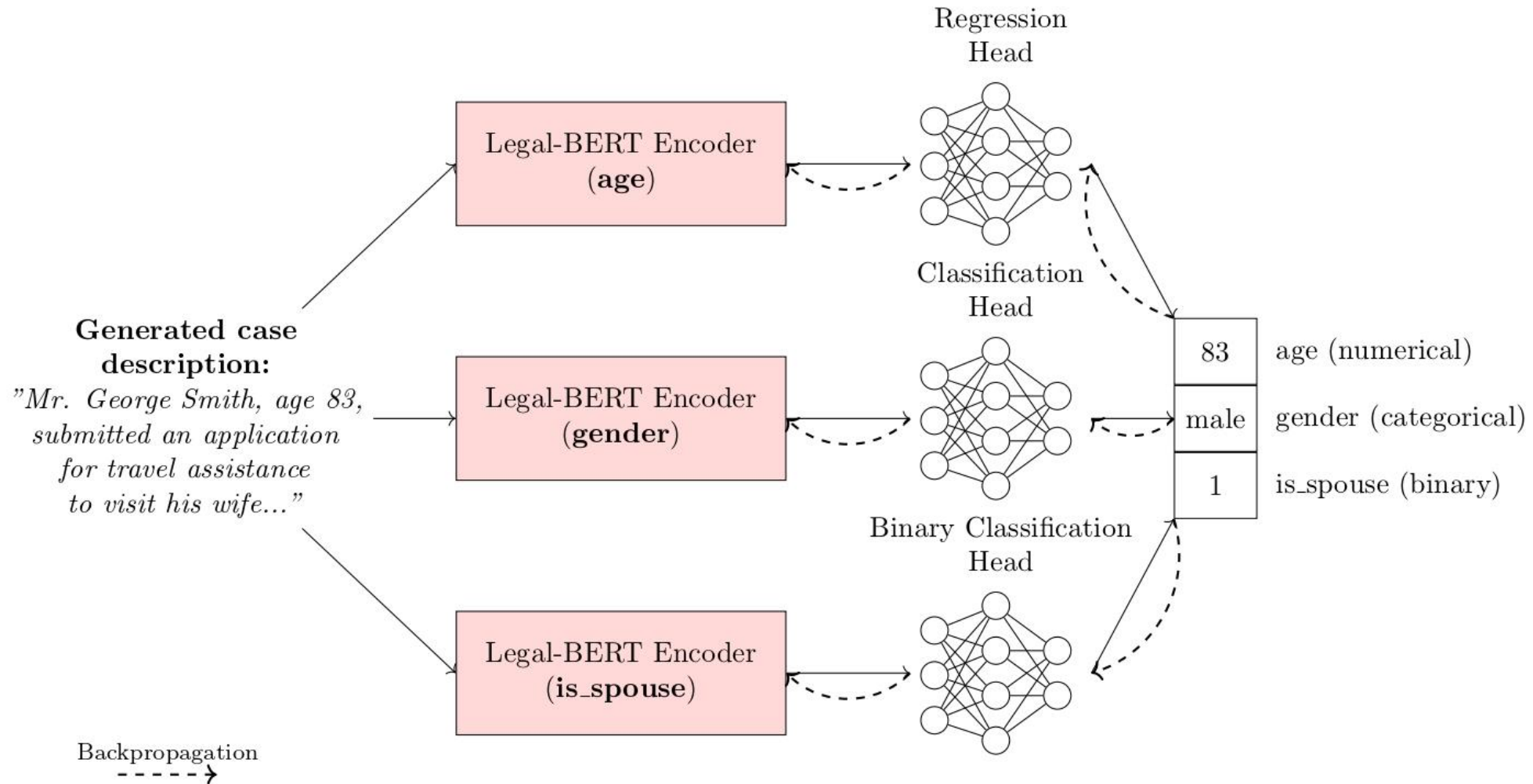
epistemic uncertainty (incomplete data)

Neuro-Argumentative Learning with Legal Text

Neuro-Argumentative Learning for Legal Reasoning



Neural Feature Extractor



Conclusions

- ABA Learning as a versatile form of logic-based/symbolic learning
 - Non-propositional variants?
- Neuro-symbolic, pipeline variants for image and text classification
 - End-to-end neuro-argumentative learning?