

LEARNING & CONTESTING ASSUMPTION-BASED ARGUMENTATION FRAMEWORKS

Emanuele De Angelis, Cnr-IASI, Italy

Joint work with

Maurizio Proietti, Cnr-IASI, Italy

Francesca Toni, IMPERIAL, UK

AI³, Rende, Italy

13 September 2025

ROADMAP

- **Assumption-based Argumentation**
ABA frameworks
- **Learning**
ABA frameworks
- **Contesting**
argumentative claims
- **Redressing**
as a way of learning



ROADMAP



- **Assumption-based Argumentation**

ABA frameworks

- **Learning**

ABA frameworks

- **Contesting
argumentative claims**

- **Redressing
as a way of learning**

Rule-based systems

for non-monotonic reasoning formalisms,
can be used for **explainable AI**
by providing **arguments** for **claims**

Arguments are **structured:**
derivations built from rules
supported by **assumptions**

Rules are **defeasible** by deriving
arguments for **contraries** of assumptions

ROADMAP

- **Assumption-based Argumentation**

ABA frameworks

- **Learning**
ABA frameworks

Automated logic-based learning
of ABA frameworks from
background knowledge

+

positive & **negative** examples

- **Contesting**
argumentative claims

- **Redressing**
as a way of learning

Algorithm based on **transformation rules**,
implemented in **Answer Set Programming**

ROADMAP



- **Assumption-based Argumentation**

ABA frameworks

- **Learning**
ABA frameworks

highly desirable property
for **human-centric AI**

- **Contesting**
argumentative claims

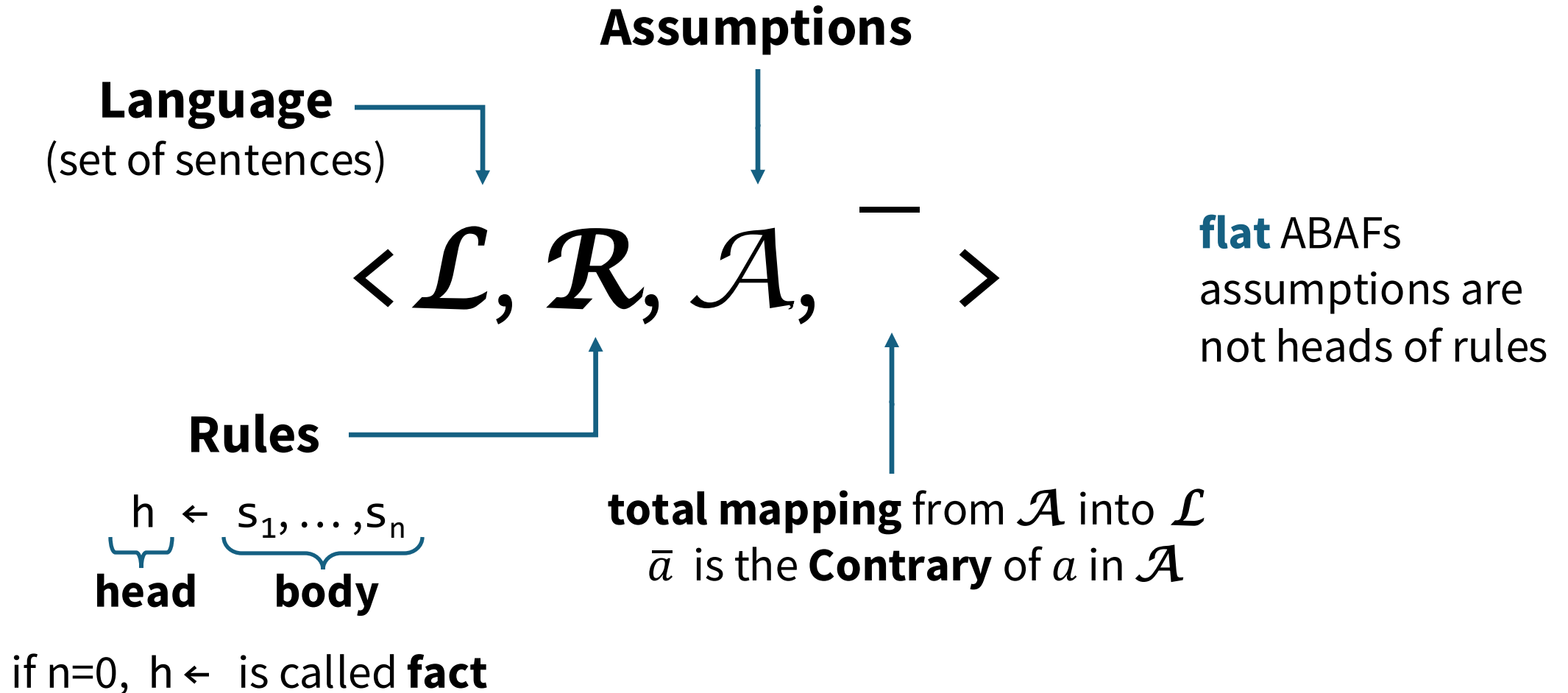
claims are subject to **contestation**:

- **rejected** claims may be **desirable**
- **accepted** claims may be **undesirable**

- **Redressing**
as a way of learning

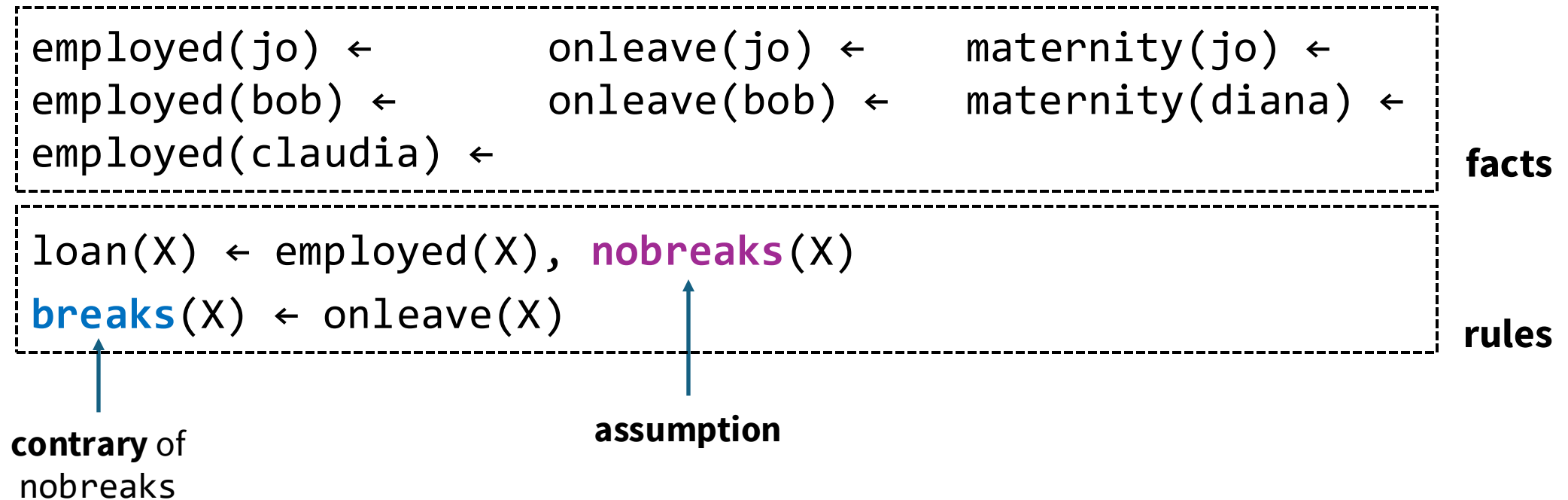
modify rules incrementally
to **reconcile** contestations

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 $\mathbf{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ (**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 $\mathbf{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}', \bar{} \rangle$ 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 $\mathbf{F} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \bar{} \rangle$ (**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 $\mathbf{F}' = \langle \mathcal{L}', \mathcal{R}', \mathcal{A}', \bar{} \rangle$ 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**

$$\langle \mathcal{L}_1, \mathcal{R}_1, \mathcal{A}_1, \neg^1 \rangle \longrightarrow \langle \mathcal{L}_2, \mathcal{R}_2, \mathcal{A}_2, \neg^2 \rangle \longrightarrow \dots \longrightarrow \langle \mathcal{L}_n, \mathcal{R}_n, \mathcal{A}_n, \neg^n \rangle$$

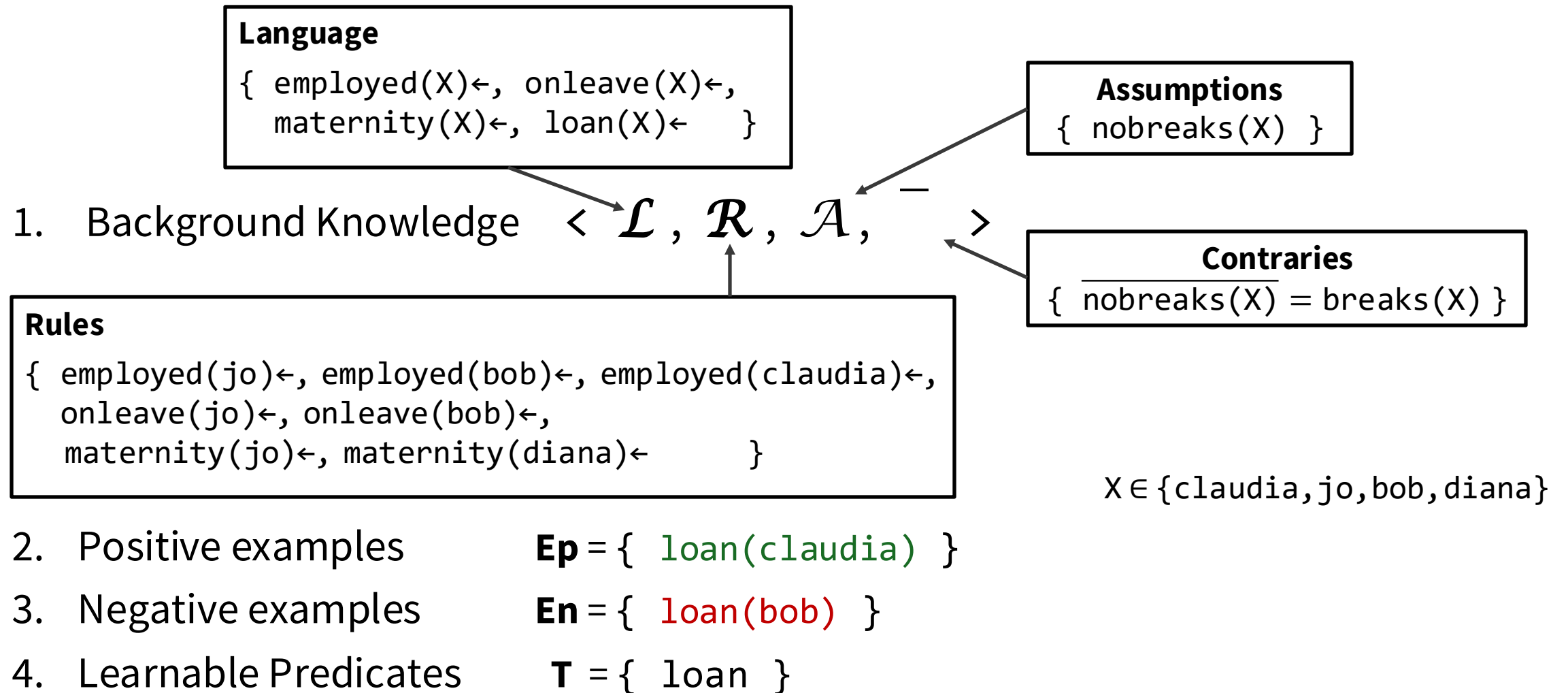
background knowledge intensional solution

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

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



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

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

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

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

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)

To get termination ...

$p1(X) \leftarrow Q(X), \text{asm_}Q(X)$

...

$p2(X) \leftarrow Q(X), \text{asm_}Q(X)$

$\text{asm_}Q(X)$
is “relative to”
 $Q(X)$

reuse

Algorithm 1: RASP-ABALearn

Input: $\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}, \mathcal{E}^+, \mathcal{E}^-, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \mathcal{T} \rangle$: redress problem
Output: $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$: incremental redress relative to $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$

```

1  $\mathcal{R} := \mathcal{R}_0$ ;  $\mathcal{A} := \mathcal{A}_0$ ;  $\overline{\phantom{x}} := \overline{\phantom{x}}^0$ ;  $\mathcal{R}_l := \emptyset$ ;
2  $RoLe()$ ;  $Gen()$ ; return  $\langle \mathcal{R}, \mathcal{A}, \overline{\phantom{x}} \rangle$ ;
3 Procedure  $RoLe()$ 
4    $P := ASP(\langle \mathcal{R}, \mathcal{A}, \overline{\phantom{x}} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \mathcal{T})$ ;
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  end
14 Procedure  $Gen()$ 
15 foreach  $\rho : (p(X) \leftarrow X=t) \in \mathcal{R}_l$  do
16    $\mathcal{R}_l := \mathcal{R}_l \setminus \{\rho\}$ ;
17   // R4. Fact Subsumption
18   if  $\neg sat(ASP(\langle \mathcal{R} \cup \mathcal{R}_l, \mathcal{A}, \overline{\phantom{x}} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$  then
19     // R2 w/ R3. Folding with Assumption Introduction
20      $\langle \rho_g, \alpha(X), C_\alpha \rangle := FoldingWasmIntro(\rho)$ ;
21      $\mathcal{R} := \mathcal{R} \cup \{\rho_g\}$ ;
22      $\mathcal{A} := \mathcal{A} \cup \{\alpha(X)\}$ ;
23      $\overline{\alpha(X)} := c\_ \alpha(X)$ ;
24     // R1. Rote Learning
25     foreach  $c\_ \alpha(t) \in C_\alpha$  do
26        $\mathcal{R}_l := \mathcal{R}_l \cup \{c\_ \alpha(X) \leftarrow X=t\}$ ;
27     end
28   end
29 end
30 Function  $FoldingWasmIntro(\rho)$ 
31 // R2. Folding
32 while  $foldable(\rho, \mathcal{R})$  do
33    $\rho := fold(\rho, \mathcal{R})$ ;
34 end
35 // R3. Assumption Introduction
36 Let  $\rho$  be  $H \leftarrow B$ ;  $X := vars(B)$ ;
37 if there exists  $\alpha(X) \in \mathcal{A}$  relative to  $B$  then
38    $\rho_g := H \leftarrow B, \alpha(X)$ ;  $C_\alpha := \emptyset$ ;
39   if  $\neg sat(ASP(\langle \mathcal{R} \cup \{\rho\}, \mathcal{A}, \overline{\phantom{x}} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$  then
40     fail;
41   end
42 else // introduce an assumption  $\alpha(X)$ , with a new predicate  $\alpha$ 
43    $\rho_g := H \leftarrow B, \alpha(X)$ ;
44    $F := \langle \mathcal{R} \cup \{\rho\}, \mathcal{A} \cup \{\alpha(X)\}, \overline{\phantom{x}} \cup \{\alpha(X) \mapsto c\_ \alpha(X)\} \rangle$ ;
45    $C_\alpha := \{c\_ \alpha(X) \mid c\_ \alpha(X) \in as(ASP(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \{c\_ \alpha\}))\}$ ;
46 end
47 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).
breaks(X) :- onleave(X).

```

```

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

```

learning facts
for contraries

- Answer sets**

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

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

- Rote learning**

```
braks(X) ← X=bob
```



SWI Prolog

+



Clingo (ASP)

Potassco

https://github.com/ABALearn/aba_asp

EXPERIMENTS

ASP-ABAllearnB

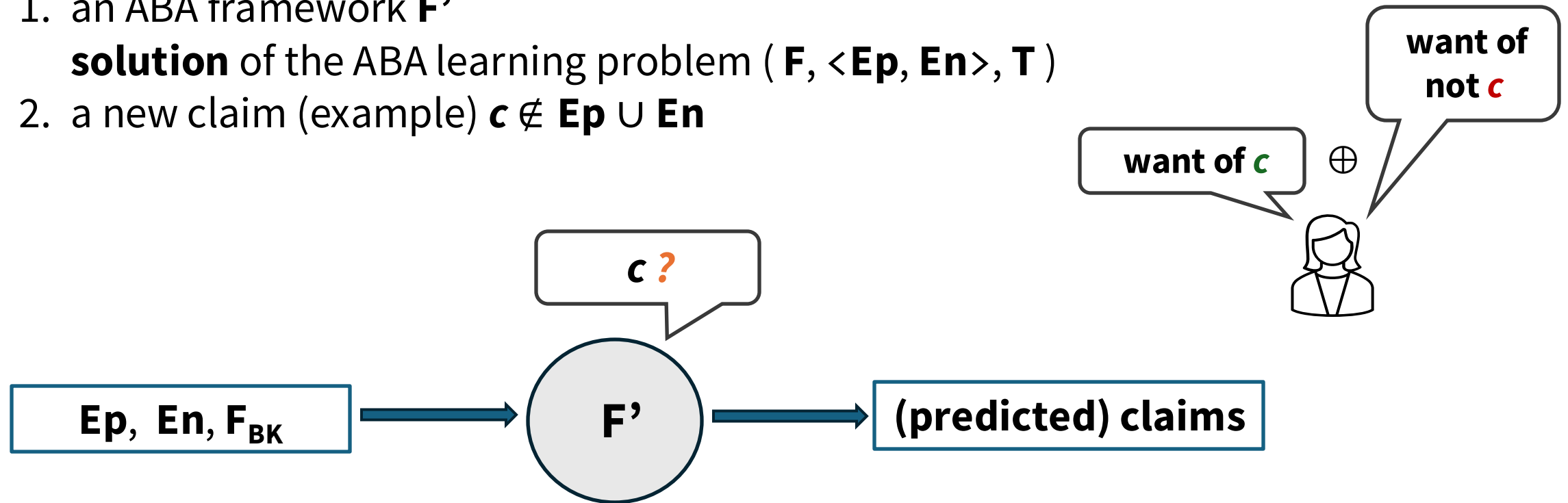
<https://doi.org/10.5281/zenodo.13330013>

Learning problem	BK	Ep	En	ASP-ABAllearnB	ILASP
Flies	8	4	2	0.01	0.09
Flies_bird&planes	10	5	2	0.02	0.25
Innocent	15	2	2	0.01	1.84
Nixon_diamond	6	1	1	0.01	unsat
Nixon_diamond_2	15	3	2	0.01	unsat
Tax_law	16	2	2	0.02	0.66
Tax_law_2	17	2	2	0.01	0.92
Acute	96	21	19	0.04	unsat
Autism	5716	189	515	23.43	timeout
Breast-w	6291	241	458	16.32	timeout

CONTESTATION

Given

1. an ABA framework \mathbf{F}'
solution of the ABA learning problem (\mathbf{F} , $\langle \mathbf{E}_p, \mathbf{E}_n \rangle$, \mathbf{T})
2. a new claim (example) $c \notin \mathbf{E}_p \cup \mathbf{E}_n$



CONTESTATION

Given

1. an ABA framework \mathbf{F}'
solution of the ABA learning problem ($F, \langle E_p, E_n \rangle, T$)
2. a new claim (example) $c \notin E_p \cup E_n$

\mathbf{F}' is **contested** by

want of c	want of not c
------------------	----------------------

iff \nexists stable extension S of \mathbf{F}' s.t.

$E_p \cup \{c\}$ are covered in S & E_n are not covered in S	E_p are covered in S & $E_n \cup \{c\}$ are not covered in S
---	---

REDRESS

When a solution F' is **contested** by either *want of c* or *want of $\text{not } c$* , **redressing** consists in deriving a **solution F''** of the ABA learning problem

- (**want of c**) $(F, \langle \mathbf{Ep} \cup \{c\}, \mathbf{En} \rangle, T)$
- (**want of $\text{not } c$**) $(F, \langle \mathbf{Ep}, \mathbf{En} \cup \{c\} \rangle, T)$

How to redress

from scratch
trivial form

1. forgetting F'
2. solving the original problem w/ c
either $(F, \langle \mathbf{Ep} \cup \{c\}, \mathbf{En} \rangle, T)$
or $(F, \langle \mathbf{Ep}, \mathbf{En} \cup \{c\} \rangle, T)$

Incremental

modifies F' *as little as possible*

1. selecting some of the **learnt** rules & making them **defeasible** by assumption introduction, to get the new problem $(\mathbf{F}'_{ai}, \langle \mathbf{Ep}, \mathbf{En} \rangle, \mathbf{T}'_{ai})$
2. solving the new problem w/ c
either $(\mathbf{F}'_{ai}, \langle \mathbf{Ep} \cup \{c\}, \mathbf{En} \rangle, \mathbf{T}'_{ai})$
or $(\mathbf{F}'_{ai}, \langle \mathbf{Ep}, \mathbf{En} \cup \{c\} \rangle, \mathbf{T}'_{ai})$

INCREMENTAL REDRESS w/ABA Learning

Suppose (the current solution) F'

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

is contested by the **want of** `loan(jo)`, which is **not covered** by any stable extension

We can **incrementally modify** F' by applying the transformation rules

→ `breaks(X) ← onleave(X), alpha(X)` (1) by **assumption introduction** rule

`c_alpha(X) ← X=jo` (2) by **rote learning** rule

`c_alpha(X) ← maternity(X)` (3) by **folding** rule

to get a new solution F'' with at least one stable extension covering `loan(jo)`

(0) Select the learnt rule to redress

EXPERIMENTS

RASP-ABALearn

https://github.com/ABALearn/aba_asp

Learning problems

six standard datasets of the UCI ML Repo as ABA learning problems

Experimental processes

Redress from scratch (**S**) vs. Incremental redress (**R**)

11 runs of **RASP-ABALearn**:

- 0**: run (S) and (R) using 90% of positive and negative examples
- 1-10**: run (S) and (R) each using a randomly selected **new example** either **positive** or **negative**

Problem		0	1	2	3	4	5	6	7	8	9	10
<i>acute</i> 495 (54, 55)	T_S	39	31	32	31	32	32	37	41	38	40	39
	T_R	36	2	3	3	2	4	7	2	3	2	3
	S_S	501	501	501	501	501	501	503	503	503	503	503
	S_R	501	501	501	501	501	501	502	502	502	502	502
<i>autism</i> 6568 (171, 464)	T_S	13524	14427	14552	15004	14985	15252	15048	15114	15246	15097	15329
	T_R	12741	970	905	999	44	1126	47	46	44	44	1144
	S_S	6953	6954	6955	6956	6956	6958	6958	6958	6958	6958	6961
	S_R	6953	6954	6955	6956	6956	6957	6957	6957	6957	6957	6958
<i>breastw</i> 6325 (216, 400)	T_S	8371	8749	9061	9258	9208	9082	9039	9086	9061	9235	9318
	T_R	8482	36	430	35	36	36	36	35	35	37	418
	S_S	6519	6519	6520	6520	6520	6520	6520	6520	6520	6520	6521
	S_R	6519	6519	6520	6520	6520	6520	6520	6520	6520	6520	6521
<i>krkp</i> 33210 (1503, 1374)	T_S	40595	42557	42250	42173	42357	41985	42417	42673	4232	4232	4232
	T_R	40475	111	1711	106	106	108	106	1682	1189	1189	1189
	S_S	33409	33409	33410	33410	33410	33410	33410	33411	33411	33411	33411
	S_R	33409	33409	33410	33410	33410	33410	33410	33411	33411	33411	33411
<i>mushroom</i> 33868 (214, 1587)	T_S	555525	559972	471200	469676	474180	552260	552583	579530	51341	51341	51341
	T_R	471191	300	11338	280	11680	11409	279	279	280	280	280
	S_S	34762	34762	34763	34763	34763	34763	34763	34763	34763	34763	34763
	S_R	34762	34762	34763	34763	34764	34765	34765	34765	34765	34765	34765
<i>voting</i> 2172 (98, 112)	T_S	663	663	675	670	669	705	707	664	664	664	664
	T_R	664	11	12	12	12	70	10	185	11	12	12
	S_S	2230	2230	2230	2230	2230	2233	2233	2229	2229	2229	2229
	S_R	2230	2230	2230	2230	2230	2231	2231	2234	2234	2234	2234

<i>autism</i> 6568 (171, 464)	T_S	13524	14427	14552	15004	14985	15252	15048	15114	15246	15097	15329
	T_R	12741	970	905	999	44	1126	47	46	44	44	1144
	S_S	6953	6954	6955	6956	6956	6958	6958	6958	6958	6958	6961
	S_R	6953	6954	6955	6956	6956	6957	6957	6957	6957	6957	6958

size of the background knowledge (# rules)

$\langle |E_p|, |E_n| \rangle$

T_S
 T_R time in ms

S_S
 S_R # of learn rules

CONCLUSIONS

- **Automatic learning of ABA frameworks** from a background knowledge, and positive and negative examples
- **Contestability** for ABA frameworks
- **Redressing as a way of learning** from additional positive or negative examples
- Experiments show
 - **folding & assumption introduction** improve **effectiveness** in learning
 - **incremental redress** is more **efficient** than **re-learning from scratch**

