LEARNING to CONTEST ARGUMENTATIVE CLAIMS

Emanuele De Angelis, Cnr-IASI, Italy

Maurizio Proietti, Cnr-IASI, Italy

Francesca Toni, IMPERIAL, UK

RuleML+RR 2025, Istanbul, Türkiye

22 September 2025

ROADMAP

- **Assumption-based Argumentation**
  ABA frameworks

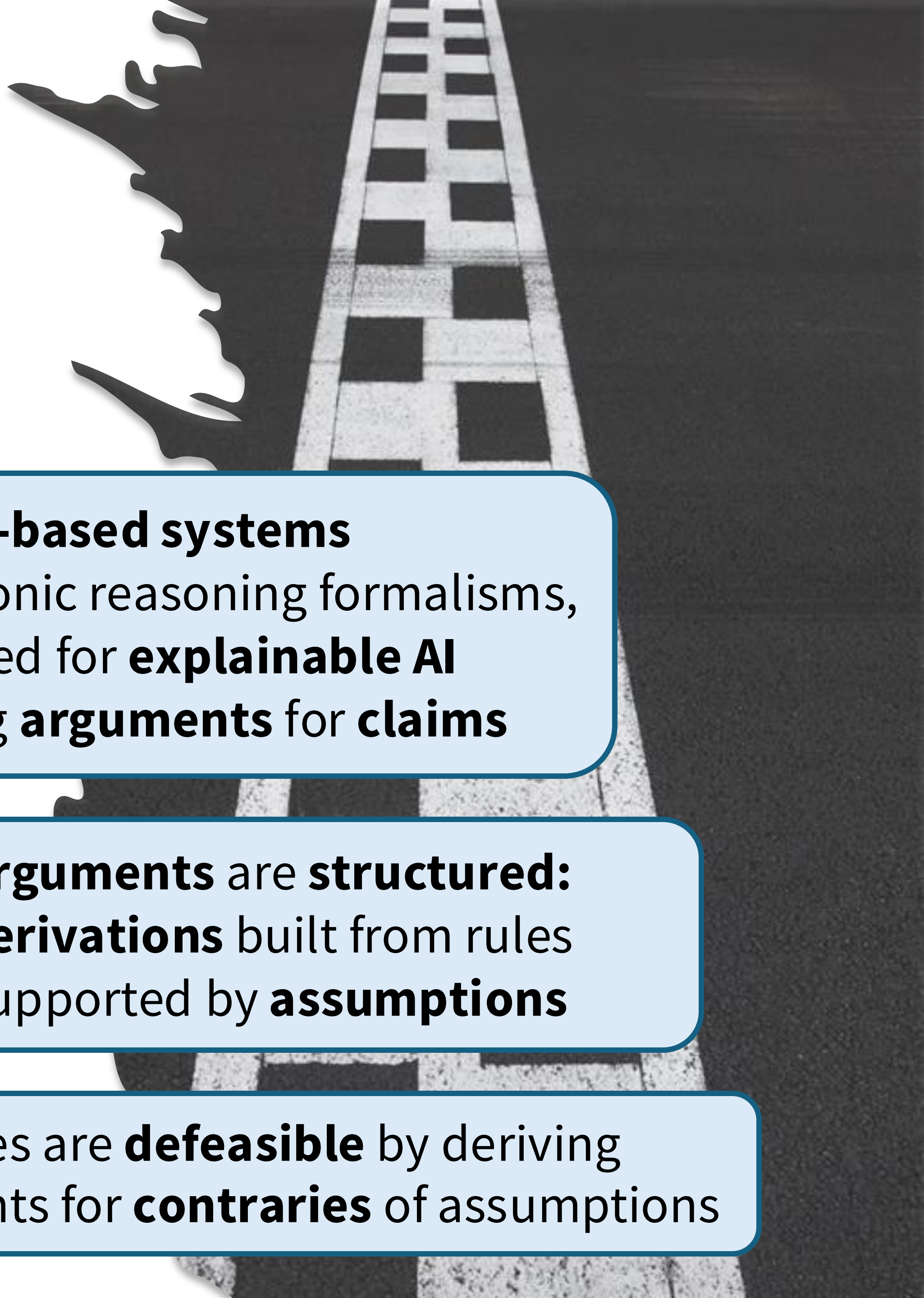
- **Learning**
  ABA frameworks

- **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**

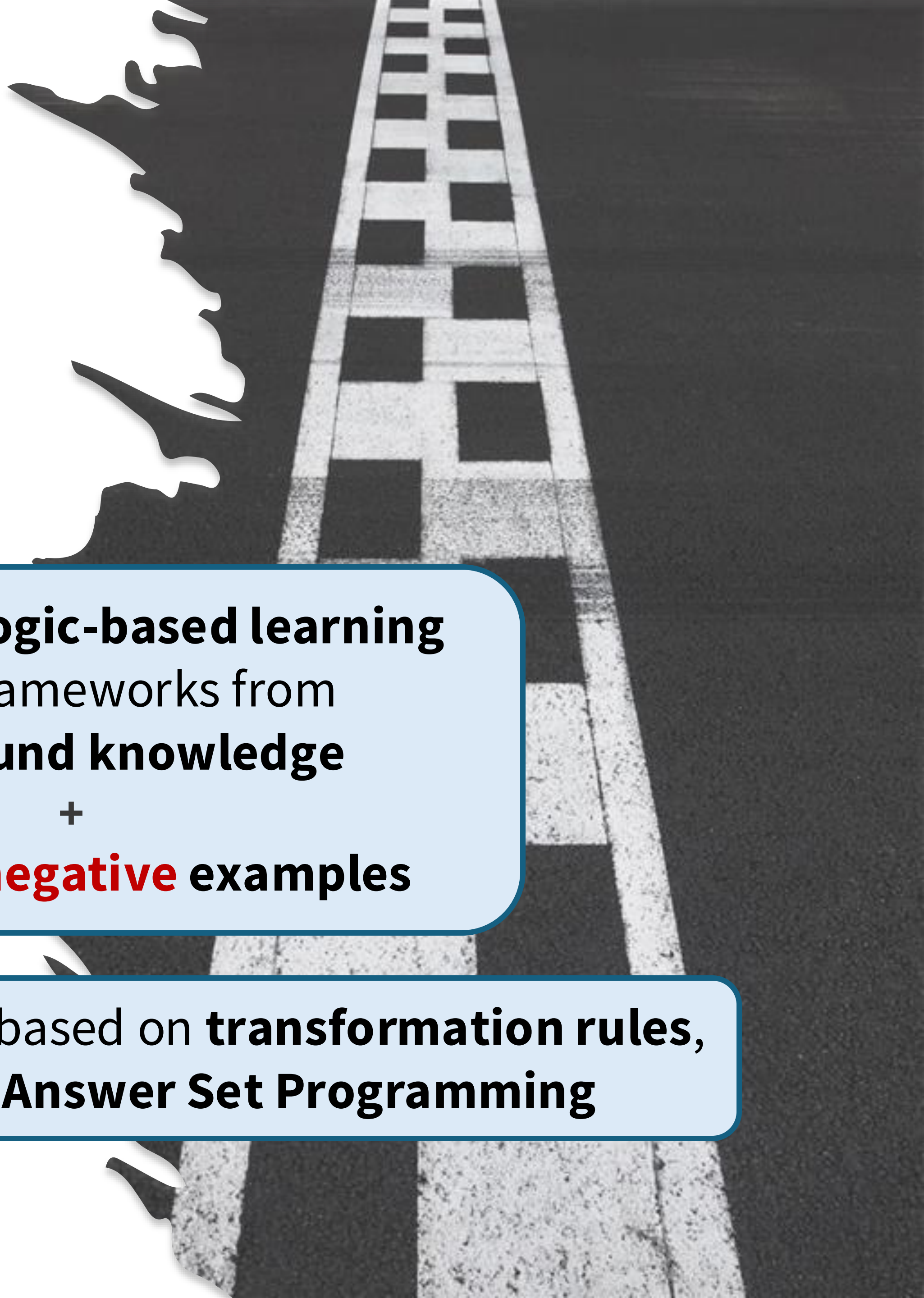
Rules are **defeasible** by deriving
arguments for **contraries** of assumptions

ROADMAP

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

Automated logic-based learning
of ABA frameworks from
background knowledge
+
positive & **negative** **examples**

Algorithm based on **transformation rules**,
using **Answer Set Programming**



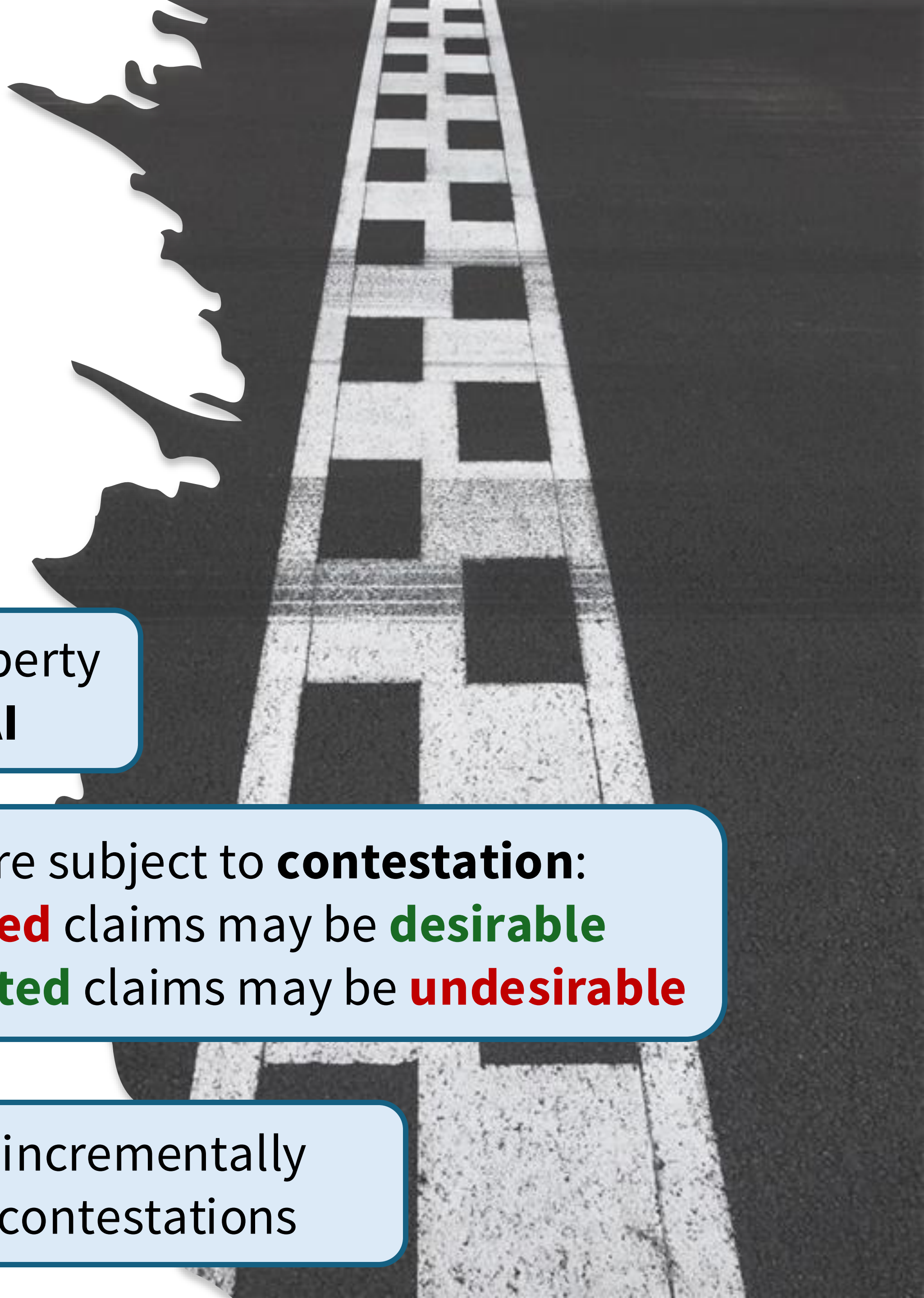
ROADMAP

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

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

claims are subject to **contestation**:
- **rejected** claims may be **desirable**
- **accepted** claims may be **undesirable**

modify rules incrementally
to **reconcile** contestations



ABA FRAMEWORKS

Language
(set of sentences)

Assumptions

$$\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$$

flat ABAFs
assumptions are
not heads of rules

Rules

$$\underbrace{h}_{\textbf{head}} \leftarrow \underbrace{s_1, \ldots, s_n}_{\textbf{body}}$$

if n=0, $h \leftarrow$ is called **fact**

total mapping from $\mathcal{A}$ into $\mathcal{L}$
$\bar{a}$ is the **Contrary** of a in $\mathcal{A}$

ABA FRAMEWORKS

an example ...

```
employed(jo) ←          onleave(jo) ←      maternity(jo) ←
employed(bob) ←         onleave(bob) ←     maternity(diana) ←
employed(claudia) ←
```

facts

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

rules

contrary of nobreaks

assumption

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)
```

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

attacks

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)

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = $< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} >$ (**background knowledge**) with at least one stable extension
2. **Ep** = { **positive examples** }
3. **En** = { **negative examples** }
4. **T** = { **learnable predicates** }

find **F’** = $< \mathcal{L}', \mathcal{R}', \mathcal{A}', \overline{} >$ with **a stable extension** S such that

i. **F** $\subseteq$ **F’**
ii. **positive** are **covered**: every positive has an argument in S
iii. **negative** are **not covered**: no negative has an argument in S

F’ is a **solution** to the ABA learning problem

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = $< \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} >$ (**background knowledge**) with at least one stable extension
2. **Ep** = { **positive examples** }
3. **En** = { **negative examples** }
4. **T** = { **lea** `loan` **es** }

find **F'** = $< \mathcal{L}', \mathcal{R}', \mathcal{A}', \overline{} >$ with **a stable extension** S such that

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

F' is a **solution** to the ABA learning problem

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = < $\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{}$ > (**background knowledge**)
   with at least one stable extension
2. **Ep** = { **positive examples** }
3. **En** = { **negative examples** }
4. **T** = { **lea**

loan

es }

find **F'** = < $\mathcal{L}', \mathcal{R}'$

```
employed(jo) ←          onleave(jo) ←     maternity(jo) ←
employed(bob) ←         onleave(bob) ←    maternity(diana) ←
employed(claudia) ←
```

i. **F ⊆ F'**

ii. **positive** ar

iii. **negative** are **not covered**: no negative has an argument in S

F' is a **solution** to the ABA learning problem

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = < $\mathcal{L}$, $\mathcal{R}$, $\mathcal{A}$, $\overline{}$ > (**background knowledge**)
   with at least one stable extension
2. **Ep** = { `loan(claudia)` **s** }
3. **En** = { **negative examples** }
4. **T** = { **lea** `loan` **es** }

find **F'** = < $\mathcal{L}'$, $\mathcal{R}'$

```
employed(jo) ←          onleave(jo) ←      maternity(jo) ←
employed(bob) ←         onleave(bob) ←     maternity(diana) ←
employed(claudia) ←
```

i. **F ⊆ F'**
ii. **positive** ar
iii. **negative** are **not covered**: no negative has an argument in S

F' is a **solution** to the ABA learning problem

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = < $\mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{}$ > (**background knowledge**)
   with at least one stable extension
2. **Ep** = { loan(claudia) **s** }
3. **En** = { **ne** loan(bob) **es** }
4. **T** = { **lea** loan **es** }

find **F'** = < $\mathcal{L}', \mathcal{R}'$

```
employed(jo) ←          onleave(jo) ←     maternity(jo) ←
employed(bob) ←         onleave(bob) ←    maternity(diana) ←
employed(claudia) ←
```

i. **F ⊆ F'**
ii. **positive** ar
iii. **negative** are **not covered**: no negative has an argument in S

F' is a **solution** to the ABA learning problem

ABA LEARNING PROBLEM

Given

1. ABA framework **F** = < $\mathcal{L}$, $\mathcal{R}$, $\mathcal{A}$, $\overline{}$ > (**background knowledge**)
   with at least one stable extension
2. **Ep** = { loan(claudia) es }
3. **En** = { ne loan(bob) es }
4. **T** = { lea loan es }

find **F'** = < $\mathcal{L}$', $\mathcal{R}$'

i. **F** ⊆ **F'**
ii. **positive** a
iii. **negative** a

```
employed(jo) ←          onleave(jo) ←       maternity(jo) ←
employed(bob) ←         onleave(bob) ←      maternity(diana) ←
employed(claudia) ←
```

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

F' is a **solution** to the ABA learning problem

ABA LEARNING
VIA TRANSFORMATION RULES

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

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

background knowledge **intensional solution**

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

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

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

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

`loan(X) ← X=claudia`

Ep = { `loan(claudia)` }

to get

$\mathcal{R}' = \mathcal{R} \cup \{$ `loan(X) ← X=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, sucha 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

```
loan(X) ← employed(X), 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!

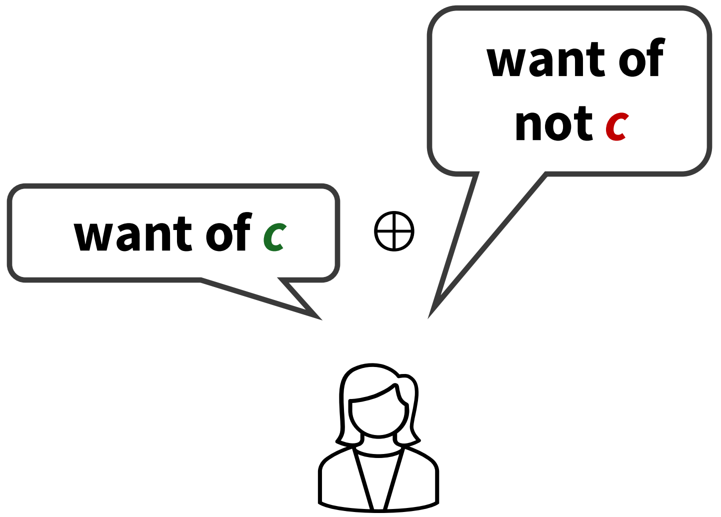
```
employed(jo) ←          onleave(jo) ←      maternity(jo) ←
employed(bob) ←         onleave(bob) ←     maternity(diana) ←
employed(claudia) ←
```

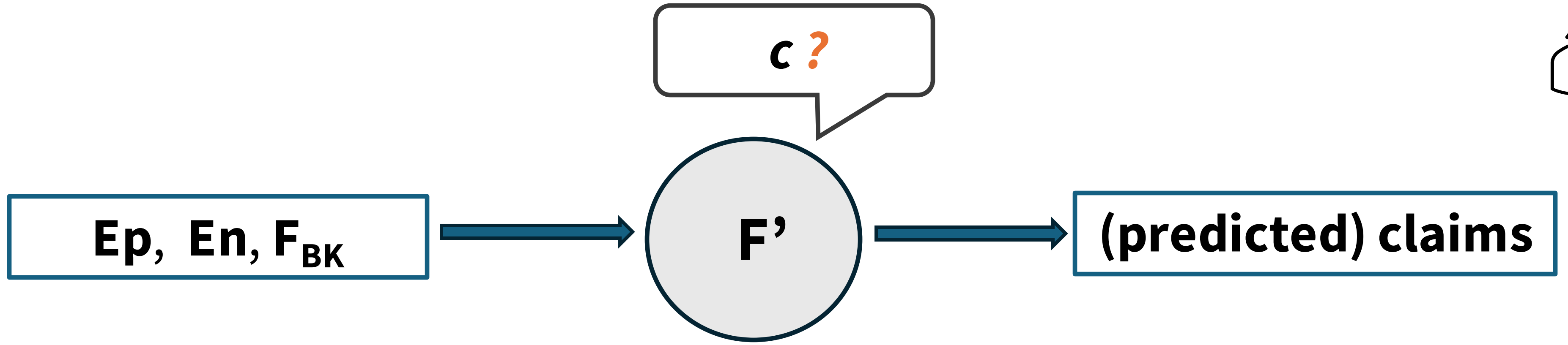
```
loan(X) ← employed(X), nobreaks(X)
breaks(X) ← onleave(X)
```

CONTESTATION

Given

1. an ABA framework **F’**
   solution of the ABA learning problem (**F**, <**Ep**, **En**>, **T**)
2. a new claim (example) $\boldsymbol{c} \notin \mathbf{Ep} \cup \mathbf{En}$





CONTESTATION

Given

1. an ABA framework **F'**
   solution of the ABA learning problem (F, <Ep, En>, T)
2. a new claim (example) ***c*** $\notin$ Ep $\cup$ En

F' is **contested** by

want of *c*	**want of *not c***

iff $\nexists$ stable extension S of **F'** s.t.

Ep $\cup$ { *c* } are **covered** in S & En are not covered in S	Ep are covered in S & En $\cup$ { *c* } are **not covered** in S

REDRESS

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

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

How to redress

from scratch

trivial form

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

Incremental

modifies F' *as little as possible*

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

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

(0) Select the learnt rule to redress

`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)`

Algorithm 1: *RASP-ABAlearn*

Input: $(\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}^{0} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \mathcal{T})$: redress problem
Output: $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$: incremental redress relative to $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$

$\mathcal{R} := \mathcal{R}_0$; $\mathcal{A} := \mathcal{A}_0$; $\overline{} := \overline{}^{0}$; $\mathcal{R}_l := \emptyset$;
RoLe(); *Gen*(); **return** $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$;
Procedure *RoLe*()
  $P := ASP(\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \mathcal{T})$;
  if $\neg sat(P)$ **then**
    fail;
  else
    $S := as(P)$;
    // R1. Rote Learning
    foreach newp(t) $\in S$ **do**
      $\mathcal{R}_l := \mathcal{R}_l \cup \{p(X) \leftarrow X = t\}$;
    end
  end
Procedure *Gen*()
  foreach $\rho : (p(X) \leftarrow X = t) \in \mathcal{R}_l$ **do**
    $\mathcal{R}_l := \mathcal{R}_l \setminus \{\rho\}$;
    // R4. Fact Subsumption
    if $\neg\, sat(ASP(\langle \mathcal{R} \cup \mathcal{R}_l, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$ **then**
      // R2 w/ R3. Folding with Assumption Introduction
      $\langle \rho_g, \alpha(X), C_\alpha \rangle := FoldingWAsmIntro(\rho)$;
      $\mathcal{R} := \mathcal{R} \cup \{\rho_g\}$;
      $\mathcal{A} := \mathcal{A} \cup \{\alpha(X)\}$;
      $\overline{\alpha(X)} := c\_\alpha(X)$;
      // R1. Rote Learning
      foreach c_α(t) $\in C_\alpha$ **do**
        $\mathcal{R}_l := \mathcal{R}_l \cup \{c\_\alpha(X) \leftarrow X = t\}$;
      end
    end
  end
Function *FoldingWAsmIntro*(ρ)
  // R2. Folding
  while $foldable(\rho, \mathcal{R})$ **do**
    $\rho := fold(\rho, \mathcal{R})$;
  end
  // R3. Assumption Introduction
  Let ρ be $H \leftarrow B$; $X := vars(B)$;
  if *there exists* $\alpha(X) \in \mathcal{A}$ *relative to* B **then**
    $\rho_g := H \leftarrow B, \alpha(X)$; $C_\alpha := \emptyset$;
    if $\neg sat(ASP(\langle \mathcal{R} \cup \{\rho\}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \emptyset))$ **then**
      fail;
    end
  else // introduce an assumption $\alpha(X)$, with a new predicate α
    $\rho_g := H \leftarrow B, \alpha(X)$;
    $F := \langle \mathcal{R} \cup \{\rho\}, \mathcal{A} \cup \{\alpha(X)\}, \overline{} \cup \{\alpha(X) \mapsto c\_\alpha(X)\} \rangle$;
    $C_\alpha := \{c\_\alpha(X) \mid c\_\alpha(X) \in as(ASP(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle, \{c\_\alpha\}))\}$;
  end
  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: breaksP(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

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
			+	−	−	−	−	+	−	+	+	+
acute	T_S	39	31	32	31	32	32	37	41	38	40	39
495	T_R	36	2	3	3	2	4	7	2	3	2	3
$\langle 54, 55\rangle$	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
			+	+	+	−	+	−	−	−	−	+
autism	T_S	13524	14427	14552	15004	14985	15252	15048	15114	15246	15097	15329
6568	T_R	12741	970	905	999	44	1126	47	46	44	44	1144
$\langle 171, 464\rangle$	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
			+	−	−	−	−	+	−	+	+	+
breastw	T_S	8371	8749	9061	9258	9208	9082	9039	9086	9061	9235	9318
6325	T_R	8482	36	430	35	36	36	36	35	35	37	418
$\langle 216, 400\rangle$	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
			−	−	−	+	+	+	−	−	+	−
krkp	T_S	40595	42557	42250	42173	42357	41985	42417	42673	[illegible]	[illegible]	[illegible]
33210	T_R	40475	111	1711	106	106	108	106	1682	1189	[illegible]	[illegible]
$\langle 1503, 1374\rangle$	S_S	33409	33409	33410	33410	33410	33410	33410	33411	[illegible]	[illegible]	[illegible]
	S_R	33409	33409	33410	33410	33410	33410	33410	33411	[illegible]	[illegible]	[illegible]
			+	−	−	−	−	+	−	−	[illegible]	[illegible]
mushroom	T_S	555525	559972	471200	469676	474180	552260	552583	579530	[illegible]	[illegible]	[illegible]
33868	T_R	471191	300	11338	280	11680	11409	279	279	280	[illegible]	[illegible]
$\langle 214, 1587\rangle$	S_S	34762	34762	34763	34763	34764	34763	34763	34763	[illegible]	[illegible]	[illegible]
	S_R	34762	34762	34763	34763	34764	34765	34765	34765	[illegible]	[illegible]	[illegible]
			+	+	−	+	+	−	−	+	[illegible]	[illegible]
voting	T_S	663	663	675	670	669	705	707	664	664	[illegible]	[illegible]
2172	T_R	664	11	12	12	12	70	10	185	11	12	12
$\langle 98, 112\rangle$	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

			+	+	+	−	+	−	−	−	−	+
autism	T_S	13524	14427	14552	15004	14985	15252	15048	15114	15246	15097	15329
6568	T_R	12741	970	905	999	44	1126	47	46	44	44	1144
$\langle 171, 464\rangle$	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)

⟨ |Ep| , |En| ⟩

T_S, T_R time in ms

S_S, S_R # of learn rules

CONCLUSIONS

- **Contestability** for **ABA frameworks** learnt from background knowledge + positive and negative examples
- Method for **incremental redress**
- **Redressing** as **a way of learning** from additional positive or negative examples
- experiments show that **incremental redress** is more **efficient** than **re-learning from scratch**