Learning to Contest Argumentative Claims

Emanuele De Angelis[1 [0000−0002−7319−8439]],
Maurizio Proietti[1 [0000−0003−3835−4931]], and
Francesca Toni[2 [0000−0001−8194−1459]]

[1] IASI-CNR, Rome, Italy
emauele.deangelis;maurizio.proietti@iasi.cnr.it
[2] Imperial College London, UK
ft@ic.ac.uk

Abstract. Contestability is a highly desirable property for human-centric AI, ensuring that the outcomes of an AI system can be challenged, and possibly changed, when interacting with humans and/or other AI systems. In this paper we study contestability of *argumentative claims* obtained from Assumption-Based Argumentation (ABA) frameworks, a unifying formalism for various non-monotonic reasoning methods that can be used for explainable AI systems. Specifically, we focus on ABA frameworks that are learnt with *ABA Learning*, a recent approach to symbolic learning from positive and negative examples, given a background knowledge. We formally define a notion of contestation when desirable claims are rejected or undesirable claims are accepted in learnt ABA frameworks. We also show that ABA Learning can be adapted to redress issues raised by contestation so that the desirable claims are accepted and the undesirable claims are rejected. This is naturally achieved by extending the learnt ABA framework without restarting from scratch, and instead preserving as much as possible thereof by considering some of its rules defeasible. We conduct several experiments with a variety of tabular datasets to demonstrate the computational advantages of our *contestable ABA Learning* in comparison with re-learning from scratch.

Keywords: Symbolic learning · Argumentation · Contestability.

1 Introduction

As the use of AI in society grows, the need for accountability, safety, security and alignment with human values of AI models also increases. Towards these ends, contestability is perceived by several as a highly desirable property for AI [20], and a crucial functionality for human-centric AI. In a nutshell, contestability amounts to ensuring that the outcomes of AI systems can be challenged, and possibly changed, if these outcomes are deemed inadequate or inappropriate by humans and/or other AI systems. For illustration, an AI system aiding a bank manager to decide on loan applications, may suggest that a specific applicant should not be granted their request if they have had career breaks in recent years; the manager or the applicant may want to contest the AI system on unfairness grounds, if the applicant's career breaks were due to parental leave.

In this paper, we study the contestability of *argumentative claims* obtained from Assumption-Based Argumentation (ABA) frameworks [1,10,29]. ABA frameworks are systems of *rules* that generalise many non-monotonic, rule-based formalisms, including (non-stratified) logic programs with negation as failure [1,4,23]. The rules in ABA frameworks may admit *assumptions* amongst their premises. These render the rules defeasible, by means of derivations for the *contraries* of the assumptions. In this setting, an argument is simply a derivation (i.e., a deduction) of a claim (i.e., a sentence) constructed via rules.

Continuing the earlier loan illustration, an ABA framework may include, for applicant jo, rules $loan(jo) \leftarrow employed(jo), nobreaks(jo)$ and $breaks(jo) \leftarrow onleave(jo)$, where $nobreaks(jo)$ is an assumption with contrary $breaks(jo)$, as well as rules (with true premises, i.e., *facts*) $employed(jo) \leftarrow$ and $onleave(jo) \leftarrow$. To determine whether claims are accepted, arguments need to be constructed and defended against attacks, according to some ABA semantics [1,10]. In the illustration, the claim $loan(jo)$ is *not accepted* (i.e., it is *rejected*), no matter which ABA semantics is adopted: an argument for $loan(jo)$ can be constructed from the rules, but it needs to rely upon the assumption $nobreaks(jo)$, and this is attacked by an argument with claim $breaks(jo)$ which cannot be attacked.

Specifically, in this paper we focus on ABA frameworks that are learnt with *ABA Learning* [6,7,22,28], a recent approach to symbolic learning from positive and negative examples (e.g., about applicants who received or not a loan in the past), given a background knowledge (e.g., knowledge about applicants). We formally define a notion of *contestation* when given desirable claims are rejected or given undesirable claims are accepted in learnt ABA frameworks. We also define a notion of *redress* of issues raised by contestation so that given desirable claims are accepted and undesirable claims are rejected. The notions of contestation and redress take into account the positive and negative examples that led to the learnt ABA framework and that still need to be accepted and rejected, respectively. Redress can be naturally achieved by extending the learnt ABA framework without restarting from scratch, and instead preserving as much as possible thereof while considering some of its rules defeasible.

Overall, we make the following contributions: (1) we define novel notions of contestation and redress in the context of ABA Learning (Section 3); (2) we define algorithmic counterparts of these notions, based on a modification of the forms of ABA Learning studied in [6,7] (Section 4); (3) we implement our algorithms as part of the ABALearn tool; and (4) we conduct several experiments with a variety of tabular datasets to demonstrate the computational advantages of our *contestable ABA Learning* in comparison with re-learning from scratch.

Related work. The need for contestable AI is advocated by several (e.g., see the recent survey in [20]) but only a handful of algorithmic solutions exist. Amongst these, [25] propose a novel approach to fine tune neural models when they are contested on the basis of having learnt causal dependencies deemed inappropriate by subject matter experts. Furthermore, [11] develop an argumentation-based model for contestable AI, but focusing on verification of claims in natural language with large language models and based on argumentation frameworks and

semantics of a different kind than for ABA. Both [11] and our work in this paper align with the vision of [20], also adopted by [8], that argumentation should play a crucial role in achieving contestable AI.

In [20], three forms of contestability are identified, for a given AI model M: (1) outputs by M for individual inputs are deemed undesirable, e.g., $y = M(x)$ for input x is deemed the wrong classification; (2) how M determines outputs for specific inputs is deemed undesirable, e.g., the way M uses a particular rule is deemed inappropriate; and (3) the full model M is contested without reference to any specific input, e.g., a rule in M could be the object of contestation. In this paper we focus on case (1) only, leaving the other two to future work.

Several other approaches to symbolic learning exist, besides ABA Learning that we rely upon. Some of these other approaches are also based on argumentation [3,9,14], while others [16,27,30] are based on learning exceptions to defeasible rules (via negation-as-failure) similarly to the case of ABA learning (which however uses assumptions). Other approaches to symbolic learning are based on abductive reasoning [15], which is closely related to the use of assumptions similarly to ABA, or answer set programming (ASP) [18,19,26], which is related to ABA as some forms of ASP can be mapped to ABA frameworks, and ASP can be used to determine acceptability of claims in ABA as we do in ABA Learning [6]. None of these approaches accommodates (forms of) contestability.

Amongst symbolic learning systems, IncrementalLAS [17] can be seen as accommodating a form of contestability (also of the first kind) by seeing learning as an incremental process. A formal and empirical comparison between our approach and IncrementalLAS requires a formal mapping between their learning and contestability problems and is thus left to future work.

2 Background

2.1 Assumption-based argumentation (ABA)

An *ABA framework* (as originally proposed in [1], but presented here following [10,29] and [4]) is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ such that:

- $\langle \mathcal{L}, \mathcal{R} \rangle$ is a deductive system, where $\mathcal{L}$ is a *language* and $\mathcal{R}$ is a set of *(inference) rules* of the form $s_0 \leftarrow s_1, \ldots, s_m$ ($m \geq 0, s_i \in \mathcal{L}$, for $1 \leq i \leq m$);
- $\mathcal{A} \subseteq \mathcal{L}$ is a (non-empty) set of *assumptions*;[3]
- $\overline{\ }$ is a *total mapping* from $\mathcal{A}$ to $\mathcal{L}$, where $\overline{a}$ is the *contrary* of a, for $a \in \mathcal{A}$ (also denoted as $\{a \mapsto \overline{a} \mid a \in \mathcal{A}\}$).

Given a rule $s_0 \leftarrow s_1, \ldots, s_m$, s_0 is the *head* and $s_1, \ldots, s_m$ is the *body*; if $m = 0$ then the rule is called a *fact* (represented as $s_0 \leftarrow$). In this paper, we focus on *flat* ABA frameworks, where assumptions are not heads of rules[4]. Elements of $\mathcal{L}$ can be any sentences, but in this paper we focus on ABA frameworks

[3] The non-emptiness requirement can always be satisfied by including in $\mathcal{A}$ a *bogus assumption*, with its own contrary, neither occurring elsewhere [29].

[4] Flat ABA frameworks of the form considered here can be mapped onto logic programs, where assumptions are replaced by the negation as failure of their contraries.

where $\mathcal{L}$ is a finite set of ground atoms. However, we will use *schemata* for rules, assumptions and contraries, using variables, similarly to logic programs, to represent compactly all instances over some underlying universe $\mathcal{U}$. In particular, we will write a fact $p(a) \leftarrow$, with a a tuple of constants, as $p(X) \leftarrow X = a$, with X a tuple of variables.

Example 1. The following ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ represents the strategy used by a bank for granting loans: a loan is approved if the applicant has been employed for a certain period without breaks. Let the universe $\mathcal{U}$ be the set $\{jo, bob, claudia, diana\}$ of constants.

$\mathcal{L} = \{loan(X), employed(X), nobreaks(X), breaks(X), onleave(X) \mid X \in \mathcal{U}\}$
$\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ where[5]
$\mathcal{R}_1 = \{\rho_1.\ employed(X) \leftarrow X = jo, \quad \rho_2.\ employed(X) \leftarrow X = bob,$
$\rho_3.\ employed(X) \leftarrow X = claudia,$
$\rho_4.\ onleave(X) \leftarrow X = jo, \quad \rho_5.\ onleave(X) \leftarrow X = bob,$
$\rho_6.\ maternity(X) \leftarrow X = jo, \quad \rho_7.\ maternity(X) \leftarrow X = diana\}$
$\mathcal{R}_2 = \{\rho_8.\ loan(X) \leftarrow employed(X), nobreaks(X),$
$\rho_9.\ breaks(X) \leftarrow onleave(X) \mid X \in \mathcal{U}\}$
$\mathcal{A} = \{nobreaks(X) \mid X \in \mathcal{U}\}$
$\overline{nobreaks(X)} = breaks(X)$, for all $X \in \mathcal{U}$.

where the assumption $nobreaks(X)$ renders rule ρ_8 defeasible: the rule can be applied only if $breaks(X)$ cannot be derived.

In the remainder, by $vars(E)$ we denote the set of variables occurring in atom, rule, or rule body E (e.g. $vars(onleave(X) \leftarrow X = jo) = \{X\}$). We will assume that variables range over the universe $\mathcal{U}$ of the individual constants occurring in $\mathcal{L}$, without, however, mentioning $\mathcal{U}$ explicitly. We will also often leave $\mathcal{L}$ implicit, and use $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ to stand for $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$.

In this paper, the semantics of (flat) ABA frameworks (to determine accepted/rejected claims) is given by *stable extensions*, defined below for arguments and attacks as follows [4,10,29]:

- *An argument for (the claim)* $s \in \mathcal{L}$ *supported by* $A \subseteq \mathcal{A}$ *and* $R \subseteq \mathcal{R}$ (denoted $A \vdash_R s$, or simply $A \vdash s$, when R is immaterial) is a finite tree with nodes labelled by sentences in $\mathcal{L}$ or by $true$, the root labelled by s, leaves either $true$ or from A, and non-leaves s' with, as children, the elements of the body of some rule in R with head s' (and all rules in R are used in the tree).
- Argument $A_1 \vdash_{R_1} s_1$ *attacks* argument $A_2 \vdash_{R_2} s_2$ iff $s_1 = \overline{a}$ for some $a \in A_2$.

Let $Args$ be the set of all arguments and $Att = \{(\beta, \gamma) \in Args \times Args \mid \beta$ attacks $\gamma\}$, for 'arguments' and 'attacks' defined as above. Then, $\Delta \subseteq Args$ is a *stable extension* iff (i) $\nexists \beta, \gamma \in \Delta$ such that $(\beta, \gamma) \in Att$ (i.e. Δ is *conflict-free*) and (ii) $\forall \gamma \in Args \setminus \Delta, \exists \beta \in \Delta$ such that $(\beta, \gamma) \in Att$ (i.e. Δ "attacks" all arguments it does not contain, thus pre-emptively "defending" itself against attacks).

We say that an ABA framework is *satisfiable* if it admits at least one stable extension, and *unsatisfiable* otherwise. We will write $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle \models_\Delta s$ to denote

[5] We use identifiers ($\rho_1, \ldots, \rho_9$ in the example) for rules, for ease of reference.

that Δ is a stable extension of $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ and $s \in \mathcal{L}$ is the claim of an argument in Δ; we also say that s is a *credulous consequence* of $\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$.

Example 2. Given the ABA framework presented in Example 1, we can construct, amongst others, the following arguments:

β_1: $\{nobreaks(jo)\} \vdash loan(jo)$
β_2: $\{nobreaks(bob)\} \vdash loan(bob)$
β_3: $\{nobreaks(claudia)\} \vdash loan(claudia)$
β_4: $\emptyset \vdash breaks(jo)$
β_5: $\emptyset \vdash breaks(bob)$

Arguments β_1, β_2 are attacked by arguments β_4, β_5, respectively. β_3 is not attacked by any argument. The unique stable extension of the ABA framework of Example 1 contains $\beta_3, \beta_4, \beta_5$, but not β_1, β_2 (and thus the claims $loan(jo)$ and $loan(bob)$ are rejected while claim $loan(claudia)$ is accepted).

2.2 ABA Learning via Transformation Rules

We define the problem of learning an ABA framework from a *background knowledge* (i.e., *any* satisfiable ABA framework), and *positive and negative examples*, following [6,7,22]. By $pred(E)$ we denote the set of predicate symbols occurring in E, where E is an atom, a rule, a set thereof, or an ABA framework.

Definition 1. *Given a satisfiable* background knowledge $F = \langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, positive examples $\mathcal{E}^+$ *and negative examples* $\mathcal{E}^-$, *with* $\mathcal{E}^+ \cup \mathcal{E}^- \subseteq \mathcal{L}$ *and* $\mathcal{E}^+ \cap \mathcal{E}^- = \emptyset$, *and a set* $\mathcal{T}$ *of* learnable *predicates, with* $\mathcal{T} \cap pred(\mathcal{A}) = \emptyset$[6] *and* $pred(\mathcal{E}^+ \cup \mathcal{E}^-) \subseteq \mathcal{T}$, *the* goal of (credulous, a.k.a. brave) ABA Learning *is to construct* $F' = \langle \mathcal{R}', \mathcal{A}', \overline{\ }' \rangle$ *such that: (i)* $\mathcal{R} \subseteq \mathcal{R}'$, *(ii) for each* $H \leftarrow B \in \mathcal{R}' \setminus \mathcal{R}$, $pred(H) \cap pred(F) \subseteq \mathcal{T}$, *(iii)* $\mathcal{A} \subseteq \mathcal{A}'$, *(iv)* $\overline{\alpha}' = \overline{\alpha}$ *for all* $\alpha \in \mathcal{A}$, *(v)* F' *is satisfiable and admits a stable extension* Δ, *such that:*

1. *for all* $e \in \mathcal{E}^+$, $F' \models_\Delta e$, *i.e., all positive examples are* covered *in* Δ
2. *for all* $e \in \mathcal{E}^-$, $F' \not\models_\Delta e$, *i.e., no negative example is* covered *in* Δ.

F' *is called a* solution based on Δ *of the* ABA Learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ *(we also say that* F' credulously entails $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$*). A solution* F' *is* intensional *when* $\mathcal{R}' \setminus \mathcal{R}$ *is made out of rule schemata without any occurrence of individual constants in the universe* $\mathcal{U}$.

Intensionality is a notion that captures the generality of a rule, as it enforces that the rule makes no explicit reference to the underlying universe.

To solve ABA learning problems, we follow an approach based on the following *transformation rules* [22].

R1. Rote Learning. Given atom $p(t) \in \mathcal{L}$, with $p \in \mathcal{T}$, add $\rho : p(X) \leftarrow X = t$ to $\mathcal{R}$. Thus, $\mathcal{R}' = \mathcal{R} \cup \{\rho\}$.

We can use R1 either to add facts from positive examples or facts for contraries of assumptions.

[6] Recall that we consider flat ABA frameworks, and thus an assumption cannot appear in the head of a learnt rule.

R2. Folding. Given distinct rules ρ_1: $H \leftarrow B_1, B_2$ and ρ_2: $K \leftarrow Eqs, B_1$, where Eqs are equalities with $vars(Eqs) \cap vars(H \leftarrow B_2) = \emptyset$, replace ρ_1 by ρ_3: $H \leftarrow Eqs, K, B_2$. Thus, $\mathcal{R}' = (\mathcal{R} \setminus \{\rho_1\}) \cup \{\rho_3\}$.

We can use R2 to generalise the body of a rule.

R3. Assumption Introduction. Replace $\rho_1 : H \leftarrow B$ in $\mathcal{R}$ by $\rho_2 : H \leftarrow B, \alpha(X)$, where X is a tuple of variables in ρ_1 and $\alpha(X)$ is a (possibly new) assumption with contrary $c\_\alpha(X)$. Thus, $\mathcal{R}' = (\mathcal{R} \setminus \{\rho_1\}) \cup \{\rho_2\}$, $\mathcal{A}' = \mathcal{A} \cup \{\alpha(X)\}$, $\overline{\alpha(X)}' = c\_\alpha(X)$, and $\overline{\beta}' = \overline{\beta}$ for all $\beta \in \mathcal{A}$.

R3 can be used to render a rule defeasible and introduces a contrary that defines the exceptions to that rule.

R4. Fact Subsumption. Let $\rho : p(X) \leftarrow X = t$ be a rule in $\mathcal{R}$ such that $\langle \mathcal{R} \setminus \{\rho\}, \mathcal{A}, \overline{\ } \rangle$ credulously entails $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$. Then, $\mathcal{R}' = \mathcal{R} \setminus \{\rho\}$.

Example 3. Let us consider, as background knowledge, $F = \langle \mathcal{R}_1, \mathcal{A}, \overline{\ } \rangle$ as per Example 1.

$\mathcal{E}^+ = \{loan(claudia)\}$ $\quad$ $\mathcal{E}^- = \{loan(bob)\}$

We consider *loan* as the unique learnable predicate, i.e., $\mathcal{T} = \{loan\}$. It can be shown that $F' = \langle \mathcal{R}_1 \cup \mathcal{R}_2, \mathcal{A}, \overline{\ } \rangle$ is an intensional solution of the ABA learning problem $\langle F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T} \rangle$. It can be seen that these two rules be derived by using the transformation rules R1–R4. In particular, rule ρ_8 can be obtained by rote learning $loan(X) \leftarrow X = claudia$ from the positive example $loan(claudia)$ (i.e., applying transformation rule R1), then folding this rule with ρ_3 (i.e., applying transformation rule R2), and finally using the assumption $nobreaks(X)$ via R3.

To support contestability, we will rely upon various algorithms and implementations of the transformation-based approach to ABA learning [6,7,28].

3 Contestation and Redress

Suppose that we have learnt an ABA framework F' from a background knowledge F, positive and negative examples $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$, and learnable predicates $\mathcal{T}$. Given a claim c, not appearing amongst the examples in $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$, we will define the contestation of F' according to the request that c is covered or not in a stable extension, say Δ, of F'. We also require that Δ continues to be a solution to the given ABA learning problem, and thus all positive examples $\mathcal{E}^+$ are covered in Δ and no negative examples in $\mathcal{E}^-$ are covered in Δ. The existential quantification on stable extensions is consistent with the credulous reasoning approach we follow in this paper.

Definition 2 (Contestation). *Let F' be a solution of an ABA learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$. Let $c \notin \mathcal{E}^+ \cup \mathcal{E}^-$ be a claim in $\mathcal{L}$ whose predicate belongs to $\mathcal{T}$. Then*

1. *F' is* contested by want of *c iff there is no stable extension Δ of F' such that (i) $\mathcal{E}^+ \cup \{c\}$ are covered in Δ, and (ii) $\mathcal{E}^-$ are not covered in Δ;*
2. *F' is* contested by want of not *c iff there is no stable extension Δ of F' such that (i) $\mathcal{E}^+$ are covered in Δ, and (ii) $\mathcal{E}^- \cup \{c\}$ are not covered in Δ.*

In the definition of contestation, the background knowledge F is not used at Points 1 and 2. However, F is relevant for the related definition of incremental redress (Definition 3) to partition the set of rules between those that can be modified (i.e., the learnt rules) and those that cannot (i.e., the rules in F).

Example 4. Let $\mathcal{E}^+ = \{p(1)\}$, $\mathcal{E}^- = \{p(2)\}$ and $c = p(3)$. Let F' admit two stable extensions, Δ_1, Δ_2 such that

$F' \models_{\Delta_1} p(1)$, $F' \not\models_{\Delta_1} p(2), p(3)$, and $F' \models_{\Delta_2} p(1), p(3)$, $F' \not\models_{\Delta_2} p(2)$.

F' is a solution based on any of the two extensions, but the want of (not) c may restrict the choice between Δ_1 and Δ_2. Indeed, F' is not contested by want of c, because of the existence of Δ_2, and F' is not contested by want of not c, because of the existence of Δ_1. If instead

$F' \models_{\Delta_1} p(1), p(3)$, $F' \not\models_{\Delta_1} p(2)$, and $F' \models_{\Delta_2} p(1), p(2)$, $F' \not\models_{\Delta_2} p(3)$,

then F' is a solution based on Δ_1 only, and thus F' is contested by want of not c, while not being contested by want of c (due to Δ_1). Finally, if

$F' \models_{\Delta_1} p(1)$, $F' \not\models_{\Delta_1} p(2), p(3)$, and $F' \models_{\Delta_2} p(3)$, $F' \not\models_{\Delta_2} p(1), p(2)$,

then, again, F' is a solution based on Δ_1 only, and thus F' is contested by want of c, while not being contested by want of not c (due to Δ_1).

Note that our choice of semantics of stable extensions enforces that, for every c, F' is either not contested, or contested by want of c, or by want of not c, but cannot be contested by want of both.

Proposition 1. *Let F' be a solution of an ABA learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$, and $c \in \mathcal{L}$. F' cannot be contested by both want of c and want of not c.*

Proof. Let F' be a solution of $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ based on stable extension Δ. Either c is covered in Δ or not. If c is covered in Δ, then all claims in $\mathcal{E}^+ \cup \{c\}$ are covered in Δ and no claim in $\mathcal{E}^-$ is covered in Δ and F' is not contested by want of c. If c is not covered in Δ, then all claims in $\mathcal{E}^+$ are covered in Δ and no claim in $\mathcal{E}^- \cup \{c\}$ is covered in Δ and F' is not contested by want of not c. □

When a solution F' of an ABA Learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ is contested, the ABA framework should be redressed to resolve the contestation. If F' is contested by want of c, with predicate in $\mathcal{T}$, then the goal of redress consists in deriving a new ABA framework F'' such that c is covered in at least one stable extension of F''. Analogously, if F' is contested by want of not c, with predicate in $\mathcal{T}$, then the goal of redress consists in deriving a new ABA framework F'' such that c is not covered in a stable extension of F''. In both cases all examples of $\mathcal{E}^+$ and $\mathcal{E}^-$ should be still be covered and not covered, respectively.

There is a trivial form of redress: we can start from the original ABA Learning problem and add c to the positive examples, in the case of want of c, or to the negative examples, in the case of want of not c. Thus, redressing reduces to forgetting F' and solving one of the two ABA Learning problems: $(F, \langle \mathcal{E}^+ \cup \{c\}, \mathcal{E}^- \rangle, \mathcal{T})$ or $(F, \langle \mathcal{E}^+, \mathcal{E}^- \cup \{c\} \rangle, \mathcal{T})$. We call this form *redress from scratch*.

Clearly, it is undesirable to redress a learnt ABA framework from scratch, if contestation is expected to happen often. In this scenario it is highly desirable to

enforce an *incremental redress*, that is, a redress that starts from F' and modifies it as little as possible.

Example 5. Let us consider the ABA framework F' that is a solution of the ABA Learning problem $\langle F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T} \rangle$ from Example 3. Suppose now that F' is contested by want of *loan*(*jo*), which is not covered by any stable extension of F'. Intuitively, one would like that the claim *loan*(*jo*) is accepted. We can incrementally modify F' by applying the transformation rules presented in Section 2.2 as follows. By R3 we introduce a new assumption $\alpha(X)$, with contrary $c\_\alpha(X)$, and transform rule ρ_9 into:

ρ_{10}. $breaks(X) \leftarrow onleave(X), \alpha(X)$

Then, by R1, we get the rule:

ρ_{11}. $c\_\alpha(X) \leftarrow X = jo$

as $c\_\alpha(jo)$ is a positive example that we want to learn. Finally, by folding ρ_{10} with ρ_6, we get

ρ_{12}. $c\_\alpha(X) \leftarrow maternity(X)$

Intuitively, the learnt rules enforce that a loan is granted to an applicant who is employed unless she/he has had a career break, excluding maternity leaves. Now *loan*(*jo*) is covered in the unique stable model of the ABA framework $F'' = \langle \mathcal{R}'', \mathcal{A}'', \overline{\ }'' \rangle$, where: $\mathcal{R}'' = \mathcal{R}_1 \cup \{\rho_8, \rho_{10}, \rho_{12}\} = \{\rho_1, \ldots, \rho_7, \rho_8, \rho_{10}, \rho_{12}\}$, $\mathcal{A}'' = \{nobreaks(X), \alpha(X)\}$, $\overline{nobreaks(X)}'' = breaks(X)$, $\overline{\alpha(X)}'' = c\_\alpha(X)$.

This example suggests that an incremental redress of a solution F' of an ABA Learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ can be realised by: (1) selecting (some of) the rules in F' that have been learnt from F and making them defeasible by assumption introduction, thus deriving $(F'_{ai}, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T}')$, where $\mathcal{T}'$ is obtained by adding the contraries of the new assumptions, and then (2) solving one of the ABA Learning problems (2.1) $(F'_{ai}, \langle \mathcal{E}^+ \cup \{c\}, \mathcal{E}^- \rangle, \mathcal{T}')$, if F' is contested by want of c, or (2.2) $(F'_{ai}, \langle \mathcal{E}^+, \mathcal{E}^- \cup \{c\} \rangle, \mathcal{T}')$, if F' is contested by want of not c.

We define incremental redress in the presence of multiple contestations.

Definition 3 (Incremental Redress). *Let $F' = \langle \mathcal{R}', \mathcal{A}', \overline{\ }' \rangle$ be a solution of an ABA Learning problem $\langle F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T} \rangle$, where $F = \langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$. Let $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$ be two sets of claims such that: (i) the predicates of $\mathcal{E}^+_C \cup \mathcal{E}^-_C$ belong to $\mathcal{T}$, and (ii) $(\mathcal{E}^+ \cup \mathcal{E}^+_C) \cap (\mathcal{E}^- \cup \mathcal{E}^-_C) = \emptyset$. Given a rule $(H \leftarrow B) \in \mathcal{R}' \setminus \mathcal{R}$, we define:*

$$(H \leftarrow B)_{ai} = \begin{cases} H \leftarrow B & \text{if an assumption } \alpha(X) \in \mathcal{A} \text{ occurs in } B \\ H \leftarrow B, \alpha(X) & \text{otherwise,} \\ & \text{where } \alpha(X) \text{ is an assumption not in } \mathcal{A}' \\ & \text{and } X = vars(H \leftarrow B), \end{cases}$$

Let F'_{ai} be $\langle \mathcal{R}'_{ai}, \mathcal{A}'_{ai}, \overline{(.)}'_{ai} \rangle$, where $\mathcal{R}'_{ai} = \mathcal{R} \cup \{\rho_{ai} \mid \rho \in \mathcal{R}' \setminus \mathcal{R}\}$, $\mathcal{A}'_{ai} = \mathcal{A}' \cup \{\alpha(X) \mid \alpha(X)$ is an assumption occurring in ρ_{ai} for some $\rho \in \mathcal{R}' \setminus \mathcal{R}\}$, and $\overline{\alpha(X)}'_{ai} = \overline{\alpha(X)}'$, for $\alpha(X) \in \mathcal{A}'$. An incremental redress *of F' with respect to $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$ is any (intensional) solution of the ABA Learning problem $(F'_{ai}, \langle (\mathcal{E}^+ \cup \mathcal{E}^+_C), (\mathcal{E}^- \cup \mathcal{E}^-_C) \rangle, \mathcal{T}'_{ai})$, where $\mathcal{T}'_{ai} = \mathcal{T}' \cup \{\overline{\alpha(X)} \mid \alpha(X) \in \mathcal{A}'_{ai}\}$.*

Example 6. Let us consider the following example, which is a variant of an example in [9]. F is a background knowledge with the following set $\mathcal{R}$ of rules:

ρ_1. $bird(X) \leftarrow X = r$, ρ_2. $bird(X) \leftarrow penguin(X)$, ρ_3. $robin(X) \leftarrow X = r$,
ρ_4. $gull(X) \leftarrow X = g$, ρ_5. $penguin(X) \leftarrow X = p1$,
ρ_6. $penguin(X) \leftarrow superpenguin(X)$, ρ_7. $superpenguin(X) \leftarrow X = p2$,
ρ_8. $ostrich(X) \leftarrow X = o$, ρ_9. $cat(X) \leftarrow X = c$, ρ_{10}. $bat(X) \leftarrow X = b$

The sets of positive and negative examples are, respectively:

$\mathcal{E}^+ = \{flies(r), flies(g)\}$ $\mathcal{E}^- = \{flies(p1), flies(c)\}$.

We consider *flies* as the unique learnable predicate, i.e., $\mathcal{T} = \{flies\}$. An intensional solution F' of the given ABA learning problem can be constructed by deriving the following two rules:

ρ_{11}. $flies(X) \leftarrow bird(X), \alpha 1(X)$ ρ_{12}. $c\_\alpha 1(X) \leftarrow penguin(X)$

Thus, the rules of the learnt framework F' are $\mathcal{R}' = \mathcal{R} \cup \{\rho_{11}, \rho_{12}\}$ (it can be shown that these two rules can be derived by using R1–R4 – see [22] for a similar derivation). Let us now assume that F is contested by want of $flies(p2)$ and $flies(b)$ and by want of not $flies(o)$. We construct F'_{ai} by assumption introduction. In particular, by R3, rule ρ_{12} is transformed into $(\rho_{12})_{ai}$, that is:

ρ_{13}. $c\_\alpha 1(X) \leftarrow penguin(X), \alpha 2(X)$

and $\mathcal{R}'_{ai} = \mathcal{R} \cup \{\rho_{11}, \rho_{13}\}$. Now, incremental redress consists in solving the new ABA Learning problem: $(F'_{ai}, \langle \mathcal{E}^+ \cup \{flies(p2), flies(b)\}, \mathcal{E}^- \cup \{flies(o), \{flies, c\_\alpha 1, c\_\alpha 2\}\})$. By R1, we learn:

ρ_{14}. $c\_\alpha 1(X) \leftarrow X = o$ ρ_{15}. $c\_\alpha 2(X) \leftarrow X = p2$ ρ_{16}. $flies(X) \leftarrow X = b$

Now, by folding, we get:

ρ_{17}. $c\_\alpha 1(X) \leftarrow ostrich(X)$ ρ_{18}. $c\_\alpha 2(X) \leftarrow superpenguin(X)$
ρ_{19}. $flies(X) \leftarrow bat(X)$.

The new ABA framework with rules $\mathcal{R} \cup \{\rho_{11}, \rho_{13}, \rho_{17}, \rho_{18}, \rho_{19}\}$ is an intensional solution of the ABA Learning problem with background knowledge F'_{ai}, and hence it is an incremental redress of F' relative to the new positive examples $\{flies(p2), flies(b)\}$ and negative examples $\{flies(\mathrm{o})\}$.

Theorem 1. *Let F'' be the incremental redress of an ABA framework F' with respect to $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$. Then, F'' is not contested by want of c, for any claim $c \in \mathcal{E}^+_C$, and F'' is not contested by want of not c, for any $c \in \mathcal{E}^-_C$.*

Proof. Directly from the definitions of solutions of an ABA Learning problem (Definition 1) and of contestation of a learnt ABA framework (Definition 2). □

It might be impossible to redress an ABA framework, simply because there are ABA Learning problems that cannnot be solved.

Example 7. The ABA Learning problem $(\langle \mathcal{R}, \mathcal{A}, \overline{\ } \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$, where: $\mathcal{R} = \{p \leftarrow q\}$; $\mathcal{A} = \emptyset$; $\mathcal{E}^+ = \{q\}$; $\mathcal{E}^- = \{p\}$; $\mathcal{T} = \{p, q\}$, has no solution.

We now show that redress from scratch of a learnt ABA framework with respect to a pair $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$ is possible if and only if incremental redress is possible. This property holds also under the further requirement that ABA frameworks are constructed by applying the transformation rules R1–R4.

Theorem 2. *Let $F' = \langle \mathcal{R}', \mathcal{A}', {}^{-\prime} \rangle$ be a solution of an ABA Learning problem $\langle F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T} \rangle$, where $F = \langle \mathcal{R}, \mathcal{A}, {}^{-} \rangle$. Let $\mathcal{E}_C^+, \mathcal{E}_C^-$ be two sets of claims such that: (i) the predicates of $\mathcal{E}_C^+ \cup \mathcal{E}_C^-$ belong to $\mathcal{T}$, and (ii) $(\mathcal{E}^+ \cup \mathcal{E}_C^+) \cap (\mathcal{E}^- \cup \mathcal{E}_C^-) = \emptyset$.*
(1. If incremental redress succeeds, then redress from scratch succeeds.)
If F'' is an incremental redress of F' with respect to $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$, then F'' is a solution of the ABA Learning problem $\langle F, \langle (\mathcal{E}^+ \cup \mathcal{E}_C^+), (\mathcal{E}^- \cup \mathcal{E}_C^-) \rangle, \mathcal{T} \rangle$. Furthermore, if F' is an intensional solution derived by R1–R4 and F'' is an intensional solution derived from F'_{ai} by R1–R4, then F'' can be derived by R1–R4.
(2. If redress from scratch succeeds, then incremental redress succeeds.)
If F'' is a solution of the ABA Learning problem $\langle F, \langle (\mathcal{E}^+ \cup \mathcal{E}_C^+), (\mathcal{E}^- \cup \mathcal{E}_C^-) \rangle, \mathcal{T} \rangle$, then there exists an incremental redress of F' with respect to $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$. Furthermore, if F'' is an intensional solution derived from F by R1–R4, then an incremental redress F''' of F' with respect to $\langle \mathcal{E}_C^+, \mathcal{E}_C^- \rangle$ can be derived by R1–R4.

Proof. (Sketch) (1) It is easy to see that if incremental redress succeeds, then redress from scratch also succeeds. Indeed, a solution of the ABA learning problem $(F'_{ai}, \langle (\mathcal{E}^+ \cup \mathcal{E}_C^+), (\mathcal{E}^- \cup \mathcal{E}_C^-) \rangle, \mathcal{T}'_{ai})$ shown in Definition 3 is also a solution of $(F, \langle (\mathcal{E}^+ \cup \mathcal{E}_C^+), (\mathcal{E}^- \cup \mathcal{E}_C^-) \rangle, \mathcal{T})$. Moreover, if F' is derived from F by applying the transformation rules R1–R4, and also the ABA framework F'' resulting from incremental redress is derived from F'_{ai} by applications of R1–R4, then F'' can be derived from F by R1–R4. Indeed, F'_{ai} is derived by applying R3 to F'. An analogous property holds if we consider intensional solutions, instead of simply solutions.
(2) We only consider that more difficult case where we use the transformation rules R1–R4 for ABA Learning. Suppose that we derive, by R1–R4, a solution F' of the ABA Learning problem $(F, \langle (\mathcal{E}^+), (\mathcal{E}^-) \rangle, \mathcal{T})$, and we also derive, by R1–R4, a solution F' of the ABA Learning problem $(F, \langle (\mathcal{E}^+ \cup \mathcal{E}_C^+), (\mathcal{E}^- \cup \mathcal{E}_C^-) \rangle, \mathcal{T})$. Then, from F', by repeated applications of R3, we can compute the ABA framework F'_{ai} with set of rules $\mathcal{R}'_{ai}$ as shown in Definition 3. Now, each rule in $\mathcal{R}'_{ai}$ is of the form $H \leftarrow B', \alpha(X)$ and, without loss of generality (by possibly renaming predicates), we can assume that $\alpha(X)$ is new assumption, that is, an assumption in $\mathcal{A}' \setminus \mathcal{A}$, whose contrary $\overline{\alpha(X)}$ does not occur in $\mathcal{R}$. We assume that (again, without loss of generality), there exists a predicate, say *dom* with a rule $dom(X) \leftarrow X = a$ for each constant a occurring in the universe $\mathcal{U}$. For each $\overline{\alpha(X)}$, by rote learning (R1) and folding (R2), we can add a rule $\overline{\alpha(X)} \leftarrow dom(X)$ and derive a new set $\mathcal{R}'_{ai} \cup \hat{\mathcal{R}}$ of rules. Let us now consider the subset $\mathcal{R}_l$ of the rules of F'' which, by hypothesis, have been derived from $\mathcal{R}$ by R1–R4. By the same sequence of applications of the transformation rules, we can derive a new ABA framework F''' with rules $\mathcal{R}_l \cup \mathcal{R}'_{ai} \cup \hat{\mathcal{R}}$. Only arguments constructed by using rules in $\mathcal{R} \cup \mathcal{R}_l$ can be accepted by a stable extension of F''', as all others would be attacked by a rule in $\hat{\mathcal{R}}$, which cannot be attacked. Thus, there is a one-to-one mapping ϕ from the stable extensions of F'' and F''' such that, for every claim c in their common language (including the examples), $F'' \models_{\Delta} c$ iff $F''' \models_{\phi(\Delta)} c$. □

Notice that, in the proof of Point (2) Theorem 2, we introduce rules $\overline{\alpha(X)} \leftarrow dom(X)$ that can be used for attacking all arguments supported by the assump-

tion $\alpha(X)$. This derivation step allows us to use, instead, the rules that, by hypothesis, can be obtained by a derivation from scratch. Obviously, this is not effective in practice, and indeed our redress algorithm of Section 4 learns suitable rules $\overline{\alpha(X)} \leftarrow p(X)$ such that $p(a)$ can be derived for a *minimal* set of constants in $\mathcal{L}$. For instance, in Example 6, we learn rules ρ_{17}, ρ_{18}, instead of rules of the form $\overline{c\_\alpha N(X)} \leftarrow \mathit{animal}(X)$, where $\mathit{animal}(X) \leftarrow X = a$ is a fact, for all constants a occurring in the language.

4 An ASP-based Algorithm for Incremental Redress

Algorithm 1 implements a strategy, called *RASP-ABAlearn*, to perform the incremental redress of a solution $F' = \langle \mathcal{R}_0, \mathcal{A}_0, {}^{-_0} \rangle$ of the ABA Learning problem $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ with respect to a pair $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$ of positive and negative examples. Algorithm 1 orchestrates the application of the transformation rules R1–R4 presented in Section 2.2 and takes advantage of a mapping between ABA frameworks under the stable extension semantics and ASP programs [2,13]. This mapping, formalised by Definition 4, reduces some reasoning tasks required by R1 and R4 to computing answer sets of an ASP program.

Definition 4. *Let* `dom(t)` *hold for all tuples* `t` *of constants of* $\mathcal{L}$. *We denote by* $ASP(\langle \mathcal{R}_0, \mathcal{A}_0, {}^{-_0} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle, \mathcal{T})$ *the following ASP program* P.

(a) Each rule in $\mathcal{R}_0$ *is a rule of* P *(rewritten in the ASP syntax)*

(b) Each $\alpha \in \mathcal{A}_0$ *is encoded in* P *by the rule* `α :- dom(X), not c_α.` , *where* `c_α` *is an ASP atom encoding* $\overline{\alpha}$, *and* $vars(\alpha) =$ `X`

(c) Each $e \in (\mathcal{E}^+ \cup \mathcal{E}^+_C)$ *is encoded in* P *as* `:- not e.`

(d) Each $e \in (\mathcal{E}^- \cup \mathcal{E}^-_C)$ *is encoded in* P *as* `:- e.`

(e) Each atom $p(X)$ *with* $p \in \mathcal{T}$ *is encoded in* P *as*

```
p(X) :- newp(X).   #minimize{1,X: newp(X)}.
```

(e.1) If $p \in pred(\overline{\alpha(X)})$ *with* $\alpha(X) \in \mathcal{A}$ *and* B *is the body in which* $\alpha(X)$ *occurs, then* P *has the choice rule* `{newp(X)} :- b.`, *where* `newp` *is a new predicate name and* `b` *is the conjunction of the non-assumption atoms in* B *such that* $vars(X) \cap vars(b) \neq \emptyset$, *and*

(e.2) If $p \in pred(\mathcal{E}^+_C)$, *then* P *has the choice rule* `{newp`(t_1)`;...;newp`(t_n)`}.`, *where* $\{newp(t_1), \ldots, newp(t_n)\} = \{newp(t) \mid p(t) \in \mathcal{E}^+_C\}$.

Point (a) is a straightforward ASP translation of the rules in $\mathcal{R}_0$. Point (b) introduces an ASP rule for each assumption in $\mathcal{A}_0$ stating that an assumption α holds if its contrary $\bar{\alpha}$ does not (i.e., any assumption holds by default). Points (c) and (d) introduce integrity constraints stating that positive examples are supported by the rules and negative examples are not. Point (e) specifies how to generate atoms that represent positive examples and contraries of assumptions. These atoms constitute the ground truth through which R1 introduces new rules into $\mathcal{R}_0$ and R4 decides to ignore examples that are already supported by rules in $\mathcal{R}_0$. In particular, the choice rules at points $(e.1)$ and $(e.2)$ generate a set of

atoms representing contraries and positive examples, respectively, and the optimization statement at point (e) enforces this set to be minimal. The optimization statement aims at reducing the number of rules required to redress a solution.

Algorithm 1 consists of two procedures: *RoLe*() and *Gen*().

RoLe() is responsible for repeatedly applying rule R1 (Rote Learning). It extends the background knowledge with a minimal set of facts to get a (non-intensional) solution to the input redress problem. *RoLe*() checks whether the ASP encoding of the learning problem P at line 4 has a solution. If P has no solution, then *RASP-ABAlearn* fails. Otherwise, it uses an answer set of P (line 8) to apply R1 (line 10). It has the same structure of *RoLe*() used in *ASP-ABAlearn*$_B$ [6], but it makes use of the new ASP encoding (Definition 4) to deal with redress.

Gen() is responsible for repeatedly applying rules R4 (Fact Subsumption), R2 (Folding), R3 (Assumption Introduction) and R1 (Rote Learning) to transform the non-intensional solution produced by *RoLe*() into an intensional solution. In contrast to *ASP-ABAlearn*$_B$ [6], it combines R2 and R3 to make the learnt rules defeasible by construction (specifically, by introducing an assumption to every rule obtained by folding). This mechanism guarantees that any solution produced by *Gen*() has the form required by F'_{ai} in Definition 3, and can therefore be used as input in a subsequent run of *RASP-ABAlearn* to redress a solution. In particular, *Gen*() takes any fact ρ introduced by *RoLe*() (line 14) and applies rule R4 to check whether it is subsumed by the rules in $\mathcal{R}_l$ (line 16). If that is not the case, it invokes *FoldingWAsmIntro*(ρ) which applies rule R2 (lines 27-29) and R3 (lines 30-40) as follows. A repeated application of R2 transforms a non intensional rule ρ into an intensional one by using the *greedy* folding strategy presented in [7]. Then, R3 introduces an assumption in the body of ρ either (i) by using an assumption in $\mathcal{A}$ introduced in a previous application of R3 or (ii) by creating a fresh new assumption to be added to $\mathcal{A}$, thereby adding the new rule ρ_g to $\mathcal{R}_l$. Finally, *Gen*() applies R1 to learn a minimal set of facts for the contrary of the assumption occurring in ρ_g.

Algorithm 1 makes also use of two subsidiary functions: (i) $as(P)$ that returns any answer set of the ASP program P, and (ii) $sat(P)$ that returns *true* if P is satisfiable (it has at least one answer set), and *false* otherwise.

By using the properties of the transformation rules R1–R4 [6], we can extend the soundness and termination results for *ASP-ABAlearn*$_B$ to the incremental redress algorithm *RASP-ABAlearn*. We omit the proofs for lack of space.

Theorem 3 (Soundness). *Let* $F' = \langle \mathcal{R}_0, \mathcal{A}_0, {}^{-0} \rangle$ *be a solution of the ABA Learning problem* $(F, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$. *If Algorithm 1 with input* $(F', \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle, \mathcal{T})$ *terminates with success, then its output is an incremental redress of* F' *with respect to* $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$. *Also, the output is an intensional ABA framework.*

Similarly to *ASP-ABAlearn*$_B$, Algorithm *RASP-ABAlearn* may terminate with failure, even if redress is possible. However, if we admit that *FoldingWAsmIntro* may return a non-intensional rule, then we get the following result.

Algorithm 1: *RASP-ABAlearn*

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

$\mathcal{R} := \mathcal{R}_0$; $\mathcal{A} := \mathcal{A}_0$; ${}^{-} := {}^{-_0}$; $\mathcal{R}_l := \emptyset$;
RoLe(); *Gen*(); **return** $\langle \mathcal{R}, \mathcal{A}, {}^{-}\rangle$;
Procedure *RoLe*()
  $P := ASP(\langle \mathcal{R}, \mathcal{A}, {}^{-}\rangle, \langle \mathcal{E}^+, \mathcal{E}^-\rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^-\rangle, \mathcal{T})$;
  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 $\neg\, sat(ASP(\langle \mathcal{R} \cup \mathcal{R}_l, \mathcal{A}, {}^{-}\rangle, \langle \mathcal{E}^+, \mathcal{E}^-\rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^-\rangle, \emptyset))$ **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 $\neg sat(ASP(\langle \mathcal{R} \cup \{\rho\}, \mathcal{A}, {}^{-}\rangle, \langle \mathcal{E}^+, \mathcal{E}^-\rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^-\rangle, \emptyset))$ **then**
      fail;
    end
  else // introduce an assumption $\alpha(X)$, with a new predicate α
    $\rho_g := H \leftarrow B, \alpha(X)$;
    $F := \langle \mathcal{R} \cup \{\rho\}, \mathcal{A} \cup \{\alpha(X)\}, {}^{-} \cup \{\alpha(X) \mapsto c\_\alpha(X)\}\rangle$;
    $C_\alpha := \{c\_\alpha(X) \mid c\_\alpha(X) \in as(ASP(F, \langle \mathcal{E}^+, \mathcal{E}^-\rangle, \langle \mathcal{E}_C^+, \mathcal{E}_C^-\rangle, \{c\_\alpha\}))\}$;
  end
  return $\langle \rho_g, \alpha(X), C_\alpha\rangle$;

Theorem 4 (Weak Completeness). *For all inputs* $(F', \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle, \mathcal{T})$, *Algorithm 1 terminates and returns a, possibly non-intensional, ABA framework, if an incremental redress of* F' *with respect to* $\langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle$ *exists.*

5 Experimental Evaluation

This section presents the experimental evaluation to assess the effectiveness and efficiency of *RASP-ABAlearn*.

Learning problems. We have formalized six ABA learning problems (reported in the first column of Table 1) from standard datasets included in the UC Irvine (UCI) Machine Learning Repository [21,30] by translating the features of each tuple into facts of the background knowledge and considering such tuple as denoting a positive or negative example according to its classification.

Implementation. We have implemented Algorithm 1 as a module of the ABALearn tool [5]. In particular, we have (i) extended ABALearn to deal with the new formalization of the learning problem, and (ii) we have implemented the *greedy* folding strategy presented in [7]. The implementation is based on the SWI-Prolog [31] system (v9.2.9) and the Clingo [12] ASP solver (v5.7.1). The tool and the datasets are available at `https://github.com/ABALearn/aba_asp`

Experimental processes. We have considered the two variants of redress presented in Section 3: (S) *Redress from scratch* and (R) *Incremental redress*. The experimental process consists in running Algorithm 1 with input $(F, \langle \emptyset, \emptyset \rangle, \langle \mathcal{E}^+_C, \mathcal{E}^-_C \rangle, \mathcal{T})$, where $\mathcal{E}^+_C$ and $\mathcal{E}^-_C$ include 90% of the tuples classified as positive and negative examples, respectively. Then, we have performed 10 additional executions of (S) and (R) each using a randomly selected new example.

Technical resources. Experiments have run on an Apple M1 with 8 GB of RAM.

Results. Table 1 shows the results of the experimental evaluation. Column 'Problem' describes the ABA learning problem: (i) the name of the problem, (ii) the size (number of facts) of the background knowledge, and (iii) the number of positive and negative examples used for the first run of Algorithm 1. The remaining columns report the results of each run of Algorithm 1: column '0' is standard ABA Learning (by setting $\langle \mathcal{E}^+, \mathcal{E}^- \rangle = \langle \emptyset, \emptyset \rangle$); columns from '1' to '10' report the results of the 10 additional runs each using a randomly selected new example. For each problem, Table 1 includes five rows: the first row gives whether the additional randomly selected example used to redress is positive or negative (columns '1'-'10'); rows 'T_S' and 'T_R' report the times in milliseconds (sum of the CPU and System time) taken by our tool to perform the experimental processes (S) and (R), respectively; rows 'S_S' and 'S_R' report the number of rules of the learnt ABA frameworks generated by performing (S) and (R), respectively.

The times demonstrate the computational advantages of performing incremental redress (S) compared to redress from scratch (R): the time to redress is always lower than the time to re-learn from scratch. Moreover, the results also show that the sizes of the learnt ABA framework are comparable, as incremental redress preserves most rules and does not add many new ones. However, in

this paper we do not present any formal result characterising the relationships between the ABA frameworks re-learnt from scratch (S) and the ones obtained by incremental redressing via *RASP-ABAlearn* (R). They could even admit different stable extensions. The only guarantee is that they are (possibly different) solutions of the same ABA learning problem, and thus each of them admits a stable extension that covers all specified positive examples and does not cover any specified negative example.

Table 1. Column 'Problem' reports: (i) the *name* of the learning problem, (ii) the size (number of facts) of the background knowledge, and (iii) the number of positive and negative examples $\langle|\mathcal{E}_C^+|, |\mathcal{E}_C^-|\rangle$ (90% of the tuples classified as positive and negative examples) used for the first run of Algorithm 1 (i.e., with input $(F, \langle\emptyset, \emptyset\rangle, \langle\mathcal{E}_C^+, \mathcal{E}_C^-\rangle, \mathcal{T})$). Column '0' reports the results of the first run. Columns from '1' to '10' report the results of the 10 additional runs each using a randomly selected new example. For each learning problem, the first row gives whether the randomly selected example to redress is positive (+) or negative (−); rows 'T_S' and 'T_R' report the times in milliseconds (sum of the CPU and System time) taken by our tool to perform a *redress from scratch* (S) and an *incremental redress* (R), respectively; rows 'S_S' and 'S_R' report the number of rules of the learnt ABA frameworks generated by performing (S) and (R), respectively.

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	42321	41987	42910
33210	T_R	40475	111	1711	106	106	108	106	1682	1189	107	106
$\langle 1503, 1374\rangle$	S_S	33409	33409	33410	33410	33410	33410	33410	33411	33412	33412	33412
	S_R	33409	33409	33410	33410	33410	33410	33410	33411	33412	33412	33412
			+	−	−	−	−	+	−	−	+	+
mushroom	T_S	555525	559972	471200	469676	474180	552260	552583	579530	513419	551619	472482
33868	T_R	471191	300	11338	280	11680	11409	279	279	280	281	279
$\langle 214, 1587\rangle$	S_S	34762	34762	34763	34763	34764	34763	34763	34763	34763	34763	34763
	S_R	34762	34762	34763	34763	34764	34765	34765	34765	34765	34765	34765
			+	+	−	+	+	−	−	+	−	+
voting	T_S	663	663	675	670	669	705	707	664	664	666	667
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

6 Conclusions

We have studied the issue of contestability for ABA frameworks learnt from a given background knowledge and sets of positive and negative examples. We have proposed a method for incremental redress when sets of claims are subject to contestation, either because one wishes to accept or reject them, in contrast to the current version of the framework. In essence, we view redressing as a way of learning from additional positive or negative examples, and hence we can use a form of ABA Learning [6,7,22] to realise it. The most important properties we use for obtaining incrementality is the ability to learn defeasible rules and to manipulate these rules through transformations. Our experiments show that incremental redress is indeed much more efficient, in terms of computation time, than re-learning from scratch, and also that the number of rules learnt incrementally is comparable with the number of rules learnt from scratch.

This work can be extended in several directions. Here we have assumed that contestation targets claims that are accepted or rejected by the learnt ABA framework, but they are consistent with the examples from which learning had been performed. We believe that our approach can be adapted to the case where new examples are in contrast to previous ones, that is, the (human or AI) agents that provide the examples may "change their mind". We could also relax the assumption that the original background knowledge is fixed, and instead allow the addition of new background knowledge together with a contestation. For instance, continuing the loan example, an applicant could support her contestation by also providing the extra fact that she owns real estate. Another interesting issue is the contestation of rules, rather than claims, as proposed in [20].

Finally, we would like to make a formal complexity analysis of the redressing problem and also perform further experimental evaluation to assess the practicality of our method. We have only considered tabular datasets, and it would be interesting to make experiments on datasets where the background knowledge consists of a set of rules, besides facts. It would also be interesting to construct a mapping between the learning problems studied here and those considered by IncrementalLAS [17], the incremental version of FastLAS [18], so as to be able to make a comparison between that system and our *RASP-ABAlearn*.

Acknowledgments. We thank support from the Royal Society, UK (IEC\R2\222045). Toni was partially funded by the ERC (grant agreement No. 101020934) and by J.P. Morgan and the RAEng, UK, under the Research Chairs Fellowships scheme (RCSRF2021\11\45). De Angelis and Proietti were supported by the MUR PRIN 2022 Project DOMAIN funded by the EU – NextGenerationEU (2022TSYYKJ, CUP B53D23013220006, PNRR, M4.C2.1.1), by the PNRR MUR project PE0000013-FAIR (CUP B53C22003630006), and by the INdAM - GNCS Project *Argomentazione Computazionale per apprendimento automatico e modellazione di sistemi intelligenti* (CUP E53C24001950001). De Angelis and Proietti are members of the INdAM-GNCS research group. Finally, we would like to thank the anonymous reviewers for their constructive remarks.

References

1. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artif. Intell. **93**, 63–101 (1997). https://doi.org/10.1016/S0004-3702(97)00015-5
2. Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (Dec 2011). https://doi.org/10.1145/2043174.2043195
3. Cocarascu, O., Stylianou, A., Cyras, K., Toni, F.: Data-empowered argumentation for dialectically explainable predictions. In: Proceedings of ECAI 2020. FAIA, vol. 325, pp. 2449–2456. IOS Press (2020). https://doi.org/10.3233/FAIA200377
4. Cyras, K., Fan, X., Schulz, C., Toni, F.: Assumption-based argumentation: Disputes, explanations, preferences. FLAP **4**(8) (2017), http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf
5. De Angelis, E., Proietti, M., Toni, F.: Code and data for "Learning Brave Assumption-Based Argumentation Frameworks via ASP". Zenodo (2024), available at https://doi.org/10.5281/zenodo.13330013
6. De Angelis, E., Proietti, M., Toni, F.: Learning brave assumption-based argumentation frameworks via ASP. In: Proceedings of ECAI 2024. FAIA, vol. 392, pp. 3445–3452. IOS Press (2024). https://doi.org/10.3233/FAIA240896
7. De Angelis, E., Proietti, M., Toni, F.: Greedy ABA learning for case-based reasoning. In: Proceedings of AAMAS 2025. p. 556–564 (2025)
8. Dignum, V., Michael, L., Nieves, J.C., Slavkovik, M., Suarez, J., Theodorou, A.: Contesting black-box AI decisions. In: Proceedings of AAMAS 2025. p. 2854–2858 (2025)
9. Dimopoulos, Y., Kakas, A.C.: Learning non-monotonic logic programs: Learning exceptions. In: Proceedings ECML 1995. pp. 122–137. LNCS 912, Springer (1995). https://doi.org/10.1007/3-540-59286-5_53
10. Dung, P., Kowalski, R., Toni, F.: Assumption-based argumentation. In: Argumentation in Artificial Intelligence, pp. 199–218. Springer (2009). https://doi.org/10.1007/978-0-387-98197-0_10
11. Freedman, G., Dejl, A., Gorur, D., Yin, X., Rago, A., Toni, F.: Argumentative large language models for explainable and contestable claim verification. In: Proceedings of AAAI-25. pp. 14930–14939. AAAI Press (2025). https://doi.org/10.1609/AAAI.V39I14.33637
12. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. TPLP **19**(1), 27–82 (2019). https://doi.org/10.1017/S1471068418000054
13. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of ICLP 1988. pp. 1070–1080. MIT Press (1988)
14. Gould, A., Paulino-Passos, G., Dadhania, S., Williams, M., Toni, F.: Preference-Based Abstract Argumentation for Case-Based Reasoning. In: Proceedings of KR 2024. pp. 394–404 (8 2024). https://doi.org/10.24963/kr.2024/37
15. Inoue, K., Haneda, H.: Learning abductive and nonmonotonic logic programs. In: Abduction and Induction: Essays on their Relation and Integration, pp. 213–231. Kluwer Academic (2000). https://doi.org/10.1007/978-94-017-0606-3_14
16. K.Inoue, Y.Kudoh: Learning extended logic programs. In: Proceedings of IJCAI 1997. pp. 176–181. Morgan Kaufmann (1997)
17. Law, M., Broda, K., Russo, A.: Search space expansion for efficient incremental inductive logic programming from streamed data. In: Proceedings of IJCAI 2022. pp. 2697–2704. ijcai.org (2022). https://doi.org/10.24963/IJCAI.2022/374

18. Law, M., Russo, A., Bertino, E., Broda, K., Lobo, J.: FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In: Proceedings of AAAI 2020. pp. 2877–2885. AAAI Press (2020). https://doi.org/10.1609/AAAI.V34I03.5678
19. Law, M., Russo, A., Broda, K.: Inductive learning of answer set programs. In: Proceedings of JELIA 2014. pp. 311–325. LNCS 8761, Springer (2014). https://doi.org/10.1007/978-3-319-11558-0_22
20. Leofante, F., Ayoobi, H., Dejl, A., Freedman, G., Gorur, D., Jiang, J., Paulino-Passos, G., Rago, A., Rapberger, A., Russo, F., Yin, X., Zhang, D., Toni, F.: Contestable AI Needs Computational Argumentation. In: Proceedings of KR 2024. pp. 888–896 (8 2024). https://doi.org/10.24963/kr.2024/83
21. Markelle Kelly, Rachel Longjohn, K.N.: The UCI machine learning repository,. https://archive.ics.uci.edu
22. Proietti, M., Toni, F.: Learning assumption-based argumentation frameworks. In: Proceedings of ILP 2022. pp. 100–116. LNCS 13779, Springer (2024). https://doi.org/10.1007/978-3-031-55630-2_8
23. Rapberger, A., Ulbricht, M., Toni, F.: On the correspondence of non-flat assumption-based argumentation and logic programming with negation as failure in the head. In: Proceedings of NMR 2024. CEUR Workshop Proceedings, vol. 3835, pp. 112–121 (2024)
24. Ray, O.: Nonmonotonic abductive inductive learning. J. Appl. Log. **7**(3), 329–340 (2009). https://doi.org/10.1016/j.jal.2008.10.007
25. Russo, F., Toni, F.: Causal discovery and knowledge injection for contestable neural networks. In: Proceedings of ECAI 2023. FAIA, vol. 372, pp. 2025–2032. IOS Press (2023). https://doi.org/10.3233/FAIA230495
26. Sakama, C.: Induction from answer sets in nonmonotonic logic programs. ACM TOCL **6**(2), 203–231 (2005). https://doi.org/10.1145/1055686.1055687
27. Shakerin, F., Salazar, E., Gupta, G.: A new algorithm to automate inductive learning of default theories. TPLP **17**(5-6), 1010–1026 (2017). https://doi.org/10.1017/S1471068417000333
28. Tirsi, C., Proietti, M., Toni, F.: ABALearn: An automated logic-based learning system for ABA frameworks. In: Proceedings of AIxIA 2023. pp. 3–16. LNCS 14318, Springer (2023). https://doi.org/10.1007/978-3-031-47546-7_1
29. Toni, F.: A tutorial on assumption-based argumentation. Argument & Computation **5**(1), 89–117 (2014). https://doi.org/10.1080/19462166.2013.869878
30. Wang, H., Shakerin, F., Gupta, G.: FOLD-RM: A scalable, efficient, and explainable inductive learning algorithm for multi-category classification of mixed data. TPLP **22**(5), 658–677 (2022). https://doi.org/10.1017/S1471068422000205
31. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. TPLP **12**(1-2), 67–96 (2012)