ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240896

Learning Brave Assumption-Based Argumentation Frameworks via ASP

Emanuele De Angelis[a,*], **Maurizio Proietti**[a] **and Francesca Toni**[b]

[a]CNR-IASI, Rome, Italy
[b]Imperial, London, UK
ORCID (Emanuele De Angelis): https://orcid.org/0000-0002-7319-8439, ORCID (Maurizio Proietti): https://orcid.org/0000-0003-3835-4931, ORCID (Francesca Toni): https://orcid.org/0000-0001-8194-1459

Abstract. Assumption-based Argumentation (ABA) is advocated as a unifying formalism for various forms of non-monotonic reasoning, including logic programming. It allows capturing defeasible knowledge, subject to argumentative debate. While, in much existing work, ABA frameworks are given up-front, in this paper we focus on the problem of automating their learning from background knowledge and positive/negative examples. Unlike prior work, we newly frame the problem in terms of brave reasoning under stable extensions for ABA. We present a novel algorithm based on transformation rules (such as Rote Learning, Folding, Assumption Introduction and Fact Subsumption) and an implementation thereof that makes use of Answer Set Programming. Finally, we compare our technique to state-of-the-art ILP systems that learn defeasible knowledge.

1 Introduction

Assumption-based Argumentation (ABA) [2, 4, 10, 30] is a form of structured argumentation broadly advocated as a unifying formalism for various formalisations of non-monotonic reasoning, including logic programming [2]. It allows capturing defeasible knowledge subject to argumentative debate, whereby arguments are deductions built from rules and supported by assumptions and, in order to be "accepted", they need to deal with attacks from other arguments (for the contraries of assumptions in their support).

In much existing work, fully-formed ABA frameworks are given up-front, e.g. to model medical guidelines [5] or planning [11]. Instead, in this paper we focus on the problem of automating their learning from background knowledge and positive and negative examples. Specifically, we consider the recent formulation of *ABA Learning* [22] for learning ABA frameworks from a background knowledge, in the form of an initial ABA framework, and positive and negative examples, in the form of sentences in the language of the background knowledge. The goal of ABA Learning is to build a larger ABA framework than the background knowledge from which arguments for all positive examples can be "accepted" and no arguments for any of the negative examples can be "accepted". In this paper, for a specific form of ABA frameworks corresponding to logic programs [2], we focus on a specific form of "acceptance", given by brave (or credulous, as commonly referred to in the argumentation literature) reasoning under the argumentation semantics of stable extensions [2, 4].

* Corresponding Author. Email: emanuele.deangelis@iasi.cnr.it

We base our approach to brave ABA Learning on transformation rules, in the spirit of [22]. We leverage on the well known correspondence [2] between stable extensions in the logic programming instance of ABA and Answer Set Programs (ASP) [13] to outline a novel implementation strategy for the form of ABA Learning we consider, pointing out along the way restrictions on ABA Learning enabling the use of ASP. We also show experimentally, on some standard benchmarks, that the resulting *ASP-ABAlearn*$_B$ system performs well in comparison with ILASP [16], a state-of-the-art system in inductive logic programming (ILP) able to learn non-stratified logic programs. In summary, our main contributions are: (i) a novel definition of *brave ABA Learning*; (ii) a novel (sound and terminating) *ASP-ABAlearn*$_B$ system for carrying out brave ABA Learning in ASP; (iii) an empirical evaluation of *ASP-ABAlearn*$_B$ showing its strengths in comparison with the ILASP system. All proofs are given in [7]. The *ASP-ABAlearn*$_B$ system is available at [8].

2 Related Work

Forms of ABA Learning have already been considered in [6, 22, 29]. Like [22] we rely upon transformation rules, adopting a variant of Subsumption and omitting to use Equality Elimination. However, we adopt a novel formulation of *brave* ABA Learning. Like [6] we use ASP as the basis for implementing ABA Learning, but, again, we focus on brave, rather than cautious, ABA Learning. Finally, [29] focuses on cautious ABA Learning and uses Python rather than ASP.

Our strategy for ABA Learning differs from other works learning argumentation frameworks, e.g. [3, 9, 21], in that it learns a different type of argumentation frameworks and, also, is based on brave reasoning rather than cautious (a.k.a. sceptical). Also, these approaches do not make use of ASP for supporting learning algorithmically.

ABA can be seen as performing abductive reasoning (as assumptions are hypotheses open for debate). Other approaches combine abductive and inductive learning [23], but they do not learn ABA frameworks. Moreover, while using a definition of abduction wrt brave/credulous stable model semantics, [23] does not identify any property of brave induction and focuses on case studies, in the context of the event calculus, with a unique answer set (where brave and cautious reasoning coincide). Some other approaches learn abductive logic programs [14], which rely upon assumptions, like ABA. A formal comparison with these methods is left for future work.

ABA captures several non-monotonic reasoning formalisms, thus

ABA Learning is related to other methods for learning non-monotonic formalisms. Some of these methods, e.g. [15, 25], do not make use of ASP, and others, e.g. [28], learn stratified logic programs with a unique stable model. Some others, e.g. [16, 26, 27], do consider ASP programs with multiple stable models. Amongst these approaches, ILASP [16] can be tailored to perform both brave and cautious induction of ASP programs, whereas [26] performs cautious induction in ASP, and the approach of [27] can perform brave induction in ASP. Induction from answer sets is also considered by Otero [19]. In this paper, the use of ASP is mainly aimed at implementing some specific tasks of our ABA Learning strategy (e.g. its use of the Rote Learning and Assumption Introduction transformation rules). More in general, differently from other learning approaches in ASP, our learning strategy is based on argumentative reasoning. A formal and empirical comparison with these methods is left for future work.

3 Background

3.1 Answer Set Programs

We use ASP [13] consisting of rules of the form

$\texttt{p :- q}_1, \ldots, \texttt{q}_k, \texttt{not q}_{k+1}, \ldots, \texttt{not q}_n$ or
$\texttt{:- q}_1, \ldots, \texttt{q}_k, \texttt{not q}_{k+1}, \ldots, \texttt{not q}_n$

where $\texttt{p}$, $\texttt{q}_1, \ldots, \texttt{q}_n$, are atoms, $\texttt{k} \geq 0$, $\texttt{n} \geq 0$, and $\texttt{not}$ denotes negation as failure. We assume that the reader is familiar with the *stable model semantics* for ASP [13], and we call *answer set* of P any set of ground atoms assigned to P by that semantics. P is said to be *satisfiable*, denoted *sat*(P), if it has an answer set, and *unsatisfiable* otherwise. An atom $\texttt{p}$ is a *brave consequence* of P if there exists an answer set A of P such that $\texttt{p} \in A$.

3.2 Assumption-Based Argumentation (ABA)

An *ABA framework* (as originally proposed in [2], but presented here following [10, 30] 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*;[1]
- $\overline{}$ is a *total mapping* from $\mathcal{A}$ into $\mathcal{L}$, 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 body is said to be *empty* (represented as $s_0 \leftarrow$ or $s_0 \leftarrow true$) and the rule is called a *fact*. In this paper we focus on *flat* ABA frameworks, where assumptions are not heads of rules. Elements of $\mathcal{L}$ can be any sentences, but in this paper we focus on ABA frameworks where $\mathcal{L}$ is a finite set of ground atoms. However, in the spirit of logic programming, we will use *schemata* for rules, assumptions and contraries, using variables to represent compactly all instances over some underlying universe. By *vars*(E) we denote the set of variables occurring in atom, rule, or rule body E.

Example 1. *We consider a variant of the well-known Nixon diamond problem [24], formalised as the ABA framework with*

$$\mathcal{L} = \{quaker(X), democrat(X), republican(X), person(X), votes\_dem(X), votes\_rep(X), normal\_quaker(X) \mid X \in \{a,b,c,d,e\}\}$$
$$\mathcal{R} = \{\rho_1 : quaker(a) \leftarrow,\ \rho_2 : quaker(b) \leftarrow,\ \rho_3 : quaker(e) \leftarrow,$$
$$\rho_4 : democrat(c) \leftarrow,\quad \rho_5 : republican(a) \leftarrow,$$
$$\rho_6 : republican(b) \leftarrow,\quad \rho_7 : republican(d) \leftarrow,$$
$$\rho_8 : democrat(X) \leftarrow person(X),\ votes\_dem(X),$$
$$\rho_9 : republican(X) \leftarrow person(X),\ votes\_rep(X),$$
$$\rho_{10} : pacifist(X) \leftarrow quaker(X), normal\_quaker(X)$$
$$\rho_{11} : person(X) \leftarrow \ \mid X \in \{a,b,c,d,e\}\}$$
$$\mathcal{A} = \{votes\_dem(X), votes\_rep(X), normal\_quaker(X) \mid X \in \{a,b,c,d,e\}\}$$
$$\overline{votes\_dem(X)} = republican(X),$$
$$\overline{votes\_rep(X)} = democrat(X),$$
$$\overline{normal\_quaker(X)} = abnormal\_quaker(X) \mid X \in \{a,b,c,d,e\}.$$

The semantics of flat ABA frameworks is given by "acceptable" extensions, i.e. sets of *arguments* able to "defend" themselves against *attacks*, in some sense, as determined by the chosen semantics. Intuitively, arguments are deductions of claims using rules and supported by assumptions, and attacks are directed at the assumptions in the support of arguments. More formally, following [4, 10, 30]:

- *An argument for (the claim) $s \in \mathcal{L}$ supported by $A \subseteq \mathcal{A}$ and $R \subseteq \mathcal{R}$* (denoted $A \vdash_R s$) is a finite tree with nodes labelled by sentences in $\mathcal{L}$ or by $true$ (a sentence not already in $\mathcal{L}$), the root labelled by s, leaves either $true$ or assumptions in 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).
- $A_1 \vdash_{R_1} s_1$ *attacks* $A_2 \vdash_{R_2} s_2$ iff $s_1 = \overline{a}$ for some $a \in A_2$.

Given a flat ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$, let $Args$ be the set of all arguments and $Att = \{(\alpha, \beta) \in Args \times Args \mid \alpha \text{ attacks } \beta\}$. Then, the notion of "acceptable" extensions we will focus on is as follows: $\Delta \subseteq Args$ is a *stable extension* iff (i) $\nexists \alpha, \beta \in \Delta$ such that $(\alpha, \beta) \in Att$ (i.e. Δ is *conflict-free*) and (ii) $\forall \beta \in Args \setminus \Delta, \exists \alpha \in \Delta$ such that $(\alpha, \beta) \in Att$ (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.

Without loss of generality, we will leave the language component of all ABA frameworks implicit, and use, e.g., $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$ to stand for $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ where $\mathcal{L}$ is the set of all sentences in $\mathcal{R}$, $\mathcal{A}$ and in the range of $\overline{}$. We will also write $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle \models_\Delta s$ to indicate that Δ is a stable extension of $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$ and $s \in \mathcal{L}$ is a claim of an argument in Δ; we also say that s is a *brave consequence* of $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$.

Example 2. *In the ABA framework $F = \langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$ from Example 1, we can construct, amongst others, the following arguments:*

$$\emptyset \vdash_{\{\rho_1\}} quaker(a) \qquad \{votes\_dem(e)\} \vdash_{\{\rho_8, \rho_{11}\}} democrat(e)$$
$$\{votes\_rep(e)\} \vdash_{\{\rho_9, \rho_{11}\}} republican(e)$$
$$\{normal\_quaker(a)\} \vdash_{\{\rho_1, \rho_{10}\}} pacifist(a)$$

with the second and third arguments attacking each other. F admits two stable extensions Δ_1 and Δ_2, where $F \models_{\Delta_1} democrat(e)$ and $F \models_{\Delta_2} republican(e)$. Also, for $i = 1, 2$, $F \models_{\Delta_i} pacifist(a)$, $F \models_{\Delta_i} pacifist(b)$, and $F \models_{\Delta_i} pacifist(e)$, as no argument for $abnormal\_quaker(X)$ can be constructed and hence ρ_{10} is applicable for every X such that $quaker(X)$ is accepted.

4 Brave ABA Learning under Stable Extensions

Here we present the instance of the ABA Learning problem that we consider in this paper. We follow the lines of [22], but we focus on a semantics based on brave consequences under stable extensions. Also, we consider a further parameter: the set $\mathcal{T}$ of predicates to be learned, which do not necessarily coincide with the predicates

[1] The non-emptiness requirement can always be satisfied by including in $\mathcal{A}$ a *bogus assumption*, with its own contrary, neither occurring elsewhere [30].

occurring in the sets of examples given in input to the learning. We will see later the role played by this set.

The *background knowledge* is *any* ABA framework $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$. *Positive* and *negative examples* are ground atoms of the form $p(c)$, for p a predicate, with arity $n \geq 0$, and c a tuple of n constants. Here, we impose that examples are non-assumptions (in the background knowledge $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$). For instance, in Example 1, *normal_quaker* cannot appear in examples. The exclusion of assumptions from examples is derived from the flatness restriction. We also assume that there is a predicate dom such that for every individual constant c, the fact $dom(c) \leftarrow$ is in $\mathcal{R}$.

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

1. *for all $e \in \mathcal{E}^+$, $\langle \mathcal{R}', \mathcal{A}', \overline{}' \rangle \models_\Delta e$,*
2. *for all $e \in \mathcal{E}^-$, $\langle \mathcal{R}', \mathcal{A}', \overline{}' \rangle \not\models_\Delta e$.*

$\langle \mathcal{R}', \mathcal{A}', \overline{}' \rangle$ is called a solution *of the* brave ABA Learning problem *$(\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$. We also say that $\langle \mathcal{R}', \mathcal{A}', \overline{}' \rangle$* bravely entails *$\langle \mathcal{E}^+, \mathcal{E}^- \rangle$. A solution is* intensional *when $\mathcal{R}' \setminus \mathcal{R}$ is made out of non-ground rule schemata.*

Condition (ii) requires that the head predicate of a learnt rule is either an "old" predicate (in the background knowledge or in the examples), in which case it needs to be a learnable predicate in $\mathcal{T}$, or a new predicate (not already in the background knowledge, e.g. the contrary of a new assumption in $\mathcal{A}' \setminus \mathcal{A}$), in which case it does not need to be in $\mathcal{T}$. Note that we only require to specify, via $\mathcal{T}$, which, amongst the predicates in the background knowledge, can be subject to learning, while imposing no restrictions on which new predicates can be learnt, unlike existing approaches, e.g., [16]. The following example illustrates the usefulness of learning new predicates.

Example 3. *Consider the background knowledge in Example 1, and*

$\mathcal{E}^+ = \{pacifist(a), pacifist(c), pacifist(e)\}$,
$\mathcal{E}^- = \{pacifist(b), pacifist(d)\}$,
$\mathcal{T} = \{pacifist, abnormal\_quaker\}$.

Solutions of $(\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ include ABA frameworks with sets of rules $\mathcal{R}'_1$ and $\mathcal{R}'_2$ whereby

$$\mathcal{R}'_1 \setminus \mathcal{R} = \{abnormal\_quaker(b) \leftarrow, \quad pacifist(c) \leftarrow\}$$
$$\mathcal{R}'_2 \setminus \mathcal{R} = \{abnormal\_quaker(X) \leftarrow republican(X),\ \alpha(X),\ c\_\alpha(X) \leftarrow quaker(X),\ normal\_quaker(X),\ pacifist(X) \leftarrow democrat(X)\}$$

with $\mathcal{A}' = \mathcal{A} \cup \{\alpha(X) \mid X \in \{a, b, c, d, e\}\}$ and $\overline{\alpha(X)}' = c\_\alpha(X)$. The new assumption $\alpha(X)$ (and its contrary) is introduced by the learning algorithm we will describe in Section 6, and can be interpreted as "X is a normal republican".

Note that intensionality in Definition 1 captures a notion of generality for the learnt rules, i.e., rules that do not make explicit reference to specific values in the universe. In the former example, the second solution can be deemed to be intensional, whereas the first is not.

The following example shows that the choice of $\mathcal{T}$ may affect the existence of a solution for the learning problem. In particular, in this example, there is no solution when $\mathcal{T}$ is the set of predicates occurring in $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$, while there is a solution by taking a larger set.

Example 4. *Consider the brave ABA Learning problem $(\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$, where: $\mathcal{R} = \{p \leftarrow q, r\}$; $\mathcal{A} = \{r\}$; $\overline{r} = p$; $\mathcal{E}^+ = \{q\}$; $\mathcal{E}^- = \emptyset$; $\mathcal{T} = \{p, q\}$. A solution for this problem is: $\mathcal{R}' = \{p \leftarrow q, r,\ p \leftarrow,\ q \leftarrow\}$; $\mathcal{A}' = \{r\}$; $\overline{r}' = p$. However, no solution exists if we take $\mathcal{T} = \{q\}$, that is, if $\mathcal{T}$ is the set of predicates occurring in $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$.*

The problem pointed out in this example may not arise if we adopt other semantics such as *grounded* or *semi-stable* extensions [4], instead of stable extensions. An alternative way to address this problem would be to assume that each rule in the background knowledge can be made defeasible, by adding an assumption in the body, whose contrary can be learnt. We leave these lines of work for future research.

5 Brave ABA Learning via Transformation Rules

To learn ABA frameworks from examples, we follow the approach based on *transformation rules* from [22], but only consider a subset of those rules: *Rote Learning*, *Folding*, *Assumption Introduction*, and (a special case of) *Subsumption*, thus ignoring *Equality Removal* [2]. Folding and Subsumption are borrowed from *logic program transformation* [20], while Rote Learning and Assumption Introduction are specific for ABA. Given an ABA framework $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$, a transformation rule constructs a new ABA framework $\langle \mathcal{R}', \mathcal{A}', \overline{}' \rangle$ (below, we will mention explicitly only the modified components).

We assume rules in $\mathcal{R}$ are written in *normalised* form as follows:

$$p_0(X_0) \leftarrow eq_1, \ldots, eq_k, p_1(X_1), \ldots, p_n(X_n)$$

where $p_i(X_i)$, for $0 \leq i \leq n$, is an atom (whose ground instances are) in $\mathcal{L}$ and eq_i, for $1 \leq i \leq k$, is an equality $t_1^i = t_2^i$, with t_j^i a term whose variables occur in the tuples $X_0, X_1, \ldots, X_n$. In particular, we represent a ground fact $p(t) \leftarrow$ as $p(X) \leftarrow X = t$. The body of a normalised rule can be freely rewritten by using the standard axioms of equality, e.g., $Y_1 = a, Y_2 = a$ can be rewritten as $Y_1 = Y_2, Y_2 = a$. For constructing arguments, we assume that, for any ABA framework, the language $\mathcal{L}$ contains all equalities between elements of the underlying universe and $\mathcal{R}$ includes all rules $a = a \leftarrow$, where a is an element of the universe. We also assume that, for all rules $H \leftarrow B \in \mathcal{R}$, $vars(H) \subseteq vars(B)$. When presenting the transformation rules, we use the following notations: (1) H, K denote heads of rules, (2) Eqs (possibly with subscripts) denotes sets of equalities, (3) B (possibly with subscripts) denotes sets of atoms.

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

We will use R1 either to add facts from positive examples or facts for contraries of assumptions, as shown by the following example.

Example 5. *Let us consider the learning problem presented in Example 3. By Rote Learning we add to $\mathcal{R}$ the following two rules:*

ρ_{12}: $abnormal\_quaker(X) \leftarrow X = b$
ρ_{13}: $pacifist(X) \leftarrow X = c$

The resulting ABA framework with rules $\mathcal{R} \cup \{\rho_{12}, \rho_{13}\}$ is a (non-intensional) solution.

We will show in Section 6 how the Rote Learning rule can be applied in an automatic way, by using ASP, so to add facts to $\mathcal{R}$

[2] The effect of equality removal can be obtained with the Folding rule R2 presented in this paper by replacing an equality $X = c$ with $dom(X)$, an atom that holds for all constants in the universe.

that allow the derivation of a, possibly non-intensional, solution of a given ABA Learning problem.

R2. Folding. Given distinct rules

ρ_1: $H \leftarrow Eqs_1, B_1, B_2$ and ρ_2: $K \leftarrow Eqs_1, Eqs_2, B_1$

with $vars(Eqs_2) \cap vars(\rho_1) = \emptyset$, replace ρ_1 by ρ_3: $H \leftarrow Eqs_2, K, B_2$. Thus, by *folding* ρ_1 *using* ρ_2, we get $\mathcal{R}' = (\mathcal{R} \setminus \{\rho_1\}) \cup \{\rho_3\}$.

Example 6. *For instance, by folding rules ρ_{12} and ρ_{13} of Example 5 using rules ρ_6 and ρ_4 (after normalisation) of Example 1, we get:*

ρ_{14}: $abnormal\_quaker(X) \leftarrow republican(X)$

ρ_{15}: $pacifist(X) \leftarrow democrat(X)$

The resulting ABA framework whose set of rules is $\mathcal{R} \cup \{\rho_{14}, \rho_{15}\}$ is no longer a solution. Indeed, pacifist(a) is not a brave consequence of $\langle \mathcal{R} \cup \{\rho_{14}, \rho_{15}\}, \mathcal{A}, \overline{} \rangle$.

Folding can be seen as a form of *inverse resolution* [18], used for generalising a rule by replacing some atoms in its body with their consequence using a rule in $\mathcal{R}$. In terms of logic program transformation [20], we can see that if we *unfold* ρ_3 wrt K using ρ_2 we get a rule more general than ρ_1. From an argumentation point of view, the following proposition shows that folding preserves arguments.

Proposition 1. *Suppose that $\mathcal{R}' = (\mathcal{R} \setminus \{\rho_1\}) \cup \{\rho_3\}$ is obtained by folding ρ_1 using ρ_2 (where ρ_1, ρ_2, ρ_3 are as in R2). Then, for any argument $A \vdash_R s$ with $R \subseteq \mathcal{R}$, there exists an argument $A \vdash_{R'} s$ with $R' \subseteq \mathcal{R}'$.*

However, folding may also introduce new arguments and new attacks, and hence we have no guarantees on the preservation of extensions, as shown by the following example.

Example 7. *Consider the ABA framework $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$, where:*

$\mathcal{R} = \{\rho_1 : \ p(X) \leftarrow q(X), r(X), \quad \rho_2 : \ s(X) \leftarrow X = a,$

$\rho_3 : \ s(X) \leftarrow q(X)\}$;

$\mathcal{A} = \{r(X)\}$; $\overline{r(X)} = p(X)$

By folding, we get $\mathcal{R}' = (\mathcal{R} \setminus \{\rho_1\}) \cup \{\rho_4 : \ p(X) \leftarrow s(X), r(X)\}$. $\mathcal{R}'$ has an extra argument $\{r(a)\} \vdash_{\{\rho_4, \rho_2\}} p(a)$, which attacks itself, because $p(a)$ is the contrary of the assumption $r(a)$, and hence the new ABA framework does not admit any stable extension.

The following Assumption Introduction transformation rule adds an assumption to the body of a rule so to make it defeasible. By learning rules for the contrary of the assumption we may be able to avoid the acceptance of unwanted arguments. This is particularly important during ABA Learning, e.g., when we want to avoid the acceptance of a negative example.

R3. Assumption Introduction. Replace $\rho_1 : H \leftarrow Eqs, B$ in $\mathcal{R}$ by $\rho_2 : H \leftarrow Eqs, B, \alpha(X)$, where X is a tuple of variables taken from $vars(\rho_1)$ 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 $\beta \in \mathcal{A}$.

Example 8. *By Assumption Introduction, from rule ρ_{14} in Example 6, we get*

ρ_{16}: $abnormal\_quaker(X) \leftarrow republican(X), \alpha(X)$

where, for $X \in \{a, b, c, d, e\}$, $\alpha(X)$ is an assumption with contrary $c\_\alpha(X)$. Now, by Rote Learning we can add the fact:

ρ_{17}: $c\_\alpha(X) \leftarrow X = a$.

The current ABA framework, is a (non-intensional) solution for the learning problem of Example 3.

The ability of Assumption Introduction, together with Rote Learning, to recover a solution after Folding, as shown in Example 8, is proved under very general conditions in the following proposition.

Proposition 2. *Suppose that $\langle \mathcal{R}_1, \mathcal{A}_1, \overline{}^1 \rangle$ is a solution of the brave ABA Learning problem $(\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}^0 \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$. Let $\mathcal{R}_2 = (\mathcal{R}_1 \setminus \{\rho_1\}) \cup \{\rho_3\}$ be obtained by folding ρ_1 using ρ_2, where ρ_1, ρ_2, and ρ_3 are as in R2. Let $\mathcal{R}_3 = (\mathcal{R}_2 \setminus \{\rho_3\}) \cup \{\rho_4\}$ be obtained by applying R3, where: $\rho_4 = H \leftarrow Eqs_2, K, B_2, \alpha(X)$, α is a new predicate symbol, and $vars(\{Eqs_1, B_1\}) \subseteq X = vars(\rho_4)$. Then there exists a set S of atoms and a set $C_\alpha = \{c\_\alpha(X) \leftarrow X = t \mid c\_\alpha(t) \in S\}$ of rules such that $\langle \mathcal{R}_3 \cup C_\alpha, \mathcal{A}_1 \cup \{\alpha(X)\}, \overline{}^1 \cup \{\alpha(X) \mapsto c\_\alpha(X)\} \rangle$ is a solution of $(\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}^0 \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$.*

The transformation rule below is a variant of Subsumption in [22].

R4. Fact Subsumption. Let $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$ be a pair of sets of positive and negative examples. Suppose that $\mathcal{R}$ contains the rule

$\rho : p(X) \leftarrow X = t$

such that $\langle \mathcal{R} \setminus \{\rho\}, \mathcal{A}, \overline{} \rangle$ bravely entails $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$. Then, by *fact subsumption relative to* $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$, we get $\mathcal{R}' = \mathcal{R} \setminus \{\rho\}$.

Example 9. *Let $\mathcal{E}^+ = \{p(a)\}$, $\mathcal{E}^- = \{p(b)\}$ and consider an ABA framework with rules*

$\mathcal{R} = \{p(X) \leftarrow q(X), r(X), \quad s(X) \leftarrow q(X), t(X),$

$p(X) \leftarrow X = a, \quad q(X) \leftarrow X = a, \quad q(X) \leftarrow X = b\}$

where $r(X)$,$t(X)$ are assumptions with $\overline{r(X)} = s(X)$ and $\overline{t(X)} = p(X)$. Then, $\langle \mathcal{R} \setminus \{p(X) \leftarrow X = a\}, \mathcal{A}, \overline{} \rangle$ bravely entails $\langle \{p(a)\}, \{p(b)\} \rangle$, and hence by Fact Subsumption, the rule $p(X) \leftarrow X = a$ can be removed from $\mathcal{R}$.

In the field of logic program transformation, the goal is to derive a new program that is *equivalent*, wrt a semantics of choice, to the initial program. Various results guarantee that, under suitable conditions, transformation rules defined in the literature, such as Unfolding and Folding, indeed enforce equivalence (e.g., wrt the least Herbrand model of definite programs [20] or the stable model semantics of normal logic programs [1]). These results have also been generalised by using argumentative notions [31].

In the context of ABA Learning, however, program equivalence is not a desirable objective, as we look for sets of rules that entail, in the sense of Definition 1, given sets of positive and negative examples. We will show in the next section how suitable sequences of applications of the transformation rules can be guided towards the goal of computing a solution of a given brave ABA Learning problem.

6 A Brave ABA Learning Algorithm

The application of the transformation rules is guided by the *ASP-ABAlearn*$_B$ algorithm (see Algorithm 1), a variant of the one in [6], which refers to a cautious stable extensions semantics. The goal of *ASP-ABAlearn*$_B$ is to derive an intensional solution for the given brave ABA Learning problem, and to achieve that goal some tasks are implemented via an ASP solver.

The *ASP-ABAlearn*$_B$ algorithm is the composition of two procedures *RoLe* and *Gen*:

(1) *RoLe* repeatedly applies Rote Learning with the objective of adding a minimal set of facts to the background knowledge $\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}^0 \rangle$ so that the new ABA framework $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$ is a (non-intensional) solution of the brave ABA Learning problem $(\langle \mathcal{R}_0, \mathcal{A}_0, \overline{}^0 \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \mathcal{T})$ given in input.

(2) *Gen* has the objective of transforming $\langle \mathcal{R}, \mathcal{A}, \overline{} \rangle$ into an intensional solution. This is done by transforming each learnt non-intensional rule as follows. First, *Gen* repeatedly applies Folding, so as to get a new intensional rule. It may happen, however, that the ABA framework with the new rule is no longer a solution of the given

brave ABA Learning problem, because, as mentioned in the previous section, new arguments and attacks may be added. In this case, *Gen* applies Assumption Introduction, followed by Rote Learning (that is, finds suitable exceptions to the learnt rules), and derives a new ABA framework that is a solution (as guaranteed by Proposition 2). Then, redundant facts are removed by Fact Subsumption. *Gen* is iterated until all learnt rules are intensional, or a failure to compute a solution is reported.

Both *RoLe* and *Gen* exploit the existence of a mapping between ABA frameworks under the stable extension semantics and ASP programs [2], and make use of the following encoding into ASP rules of a given ABA Learning problem (we use the teletype font for ASP rules).

Definition 2. *Let* $\texttt{dom(t)}$ *hold for all tuples* $\texttt{t}$ *of constants of* $\mathcal{L}$*. We denote by* $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$ *the set of ASP rules constructed as described at the following points (a)–(e).*

(a) Each rule in $\mathcal{R}$ *is rewritten in the ASP syntax (see Section 3).*
(b) Each $\alpha \in \mathcal{A}$ *occurring in* $\mathcal{R}$ *is encoded as the following ASP rule, where* $\texttt{c\_}\alpha$ *is an ASP atom encoding* $\overline{\alpha}$*, and* $vars(\alpha) = \texttt{X}$*:*
$\alpha \texttt{ :- dom(X), not c\_}\alpha.$
(c) Each $e \in \mathcal{E}^+$ *is encoded as the ASP rule* $\texttt{:- not e}$*.*
(d) Each $e \in \mathcal{E}^-$ *is encoded as the ASP rule* $\texttt{:- e}$*.*
(e) Each atom $p(X)$ *with* $p \in \mathcal{T}$ *is encoded through the following ASP rules, where* $\texttt{p}'$ *is a new predicate name:*
$\texttt{p(X) :- p}'\texttt{(X).} \qquad \texttt{\{p}'\texttt{(X)\} :- dom(X).}$
with the ASP directive $\texttt{\#minimize\{1,X : p}'\texttt{(X)\}}$*.*

The rule $\texttt{\{p}'\texttt{(X)\} :- dom(X)}$ at point (e) is a *choice rule* [12], which has an answer set for each subset of $\{\texttt{p(t)} \mid \texttt{dom(t)} \text{ holds}\}$[3]. We use predicate $\texttt{p}'$ to distinguish new facts from the atoms $\texttt{p(X)}$ which are already consequences of the ASP rules (a)–(d). The $\texttt{\#minimize\{1,X : p}'\texttt{(X)\}}$ directive does not affect satisfiability, but enforces the computation of answer sets with *minimal* subsets of $\texttt{p}'$ atoms, and hence the addition of a minimal set of new facts by *RoLe*.

The following properties of $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$ will be used for showing the soundness of the *ASP-ABAlearn*$_B$ algorithm.

Theorem 1. $\langle\mathcal{R},\mathcal{A},\overline{}\rangle$ *bravely entails* $\langle\mathcal{E}^+,\mathcal{E}^-\rangle$ *if and only if the set of rules* $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\emptyset)$ *is satisfiable.*

Thus, in particular, a claim $s \in \mathcal{L}$ is a brave consequence of $\langle\mathcal{R},\mathcal{A},\overline{}\rangle$ under stable extensions if and only if the set of rules $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\{s\},\emptyset\rangle,\emptyset)$ is satisfiable.

Theorem 2. *(1) There exists a solution of the brave learning problem* $(\langle\mathcal{R},\mathcal{A},\overline{}\rangle, \langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$ *if and only if the set of rules* $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$ *is satisfiable. (2) Suppose that* S *is an answer set of* $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$*, then* $\langle\mathcal{R}',\mathcal{A},\overline{}\rangle$ *is a solution if* $\mathcal{R}' = \mathcal{R} \cup \{p(X) \leftarrow X = t \mid p \in \mathcal{T} \text{ and } \texttt{p}'\texttt{(t)} \in S\}$*.*

Let us comment the *ASP-ABAlearn*$_B$ algorithm in some detail. At line 4 of the *RoLe* procedure, the algorithm sets to P the ASP encoding $ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$ of the input learning problem. The failure at line 6 is due to Theorem 2: if P is unsatisfiable, then the learning problem has no solutions. Otherwise, due to the

[3] In the implementation of Algorithm 1 we make use of the following optimisation, which reduces the domain size of each variable in $\texttt{X}$, and hence the size of the grounding of the choice rule. If $p \in \mathcal{T}$ is the predicate of a contrary of an assumption $\alpha(X)$ and B is the body in which $\alpha(X)$ occurs, $\texttt{dom(X)}$ is replaced by the conjunction of the non-assumption atoms b in B such that $vars(\texttt{X}) \cap vars(b) \neq \emptyset$. If p occurs in $\mathcal{E}^+$, the choice rule is simplified to "$\{\texttt{e}_1;\ldots;\texttt{e}_n\}$." where $\{e_1,\ldots,e_n\} = \{p'(t) \mid p(t) \in \mathcal{E}^+\}$.

Algorithm 1: *ASP-ABAlearn*$_B$

Input: $(\langle\mathcal{R}_0,\mathcal{A}_0,\overline{}^{\,0}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$: learning problem
Output: $\langle\mathcal{R},\mathcal{A},\overline{}\rangle$: intensional solution

1 $\mathcal{R} := \mathcal{R}_0$; $\mathcal{A} := \mathcal{A}_0$; $\overline{} := \overline{}^{\,0}$; $\mathcal{R}_l := \emptyset$;
2 *RoLe*(); *Gen*(); **return** $\langle\mathcal{R},\mathcal{A},\overline{}\rangle$;
3 **Procedure** *RoLe*()
4 $P := ASP(\langle\mathcal{R},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\mathcal{T})$;
5 **if** $\neg sat(P)$ **then**
6 fail;
7 **else**
8 $S := getAS(P)$;
/* - Rote learning - */
9 **foreach** $\texttt{p}'\texttt{(t)} \in S$ **do**
10 $\mathcal{R}_l := \mathcal{R}_l \cup \{p(X) \leftarrow X = t\}$;
11 **end**
12 **end**
13 **Procedure** *Gen*()
14 **foreach** $\rho : (p(X) \leftarrow X = t) \in \mathcal{R}_l$ **do**
15 $\mathcal{R}_l := \mathcal{R}_l \setminus \{\rho\}$; $\mathcal{R}_t := \mathcal{R} \cup \mathcal{R}_l$;
/* - Fact subsumption - */
16 **if** $\neg\, sat(ASP(\langle\mathcal{R}_t,\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\emptyset))$ **then**
/* - Folding - */
17 $\rho_f := applyFolding(\rho,\mathcal{R})$;
18 **if** $\neg\, sat(ASP(\langle\mathcal{R}_t \cup \{\rho_f\},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\emptyset))$ **then**
/* - Assumption introduction - - - - - - - - - - - - - - - - - */
19 $\langle\rho_d,\alpha(X),S\rangle := applyAsmIntro(\rho_f,\mathcal{R}_t)$;
/* (1) ρ_d is a rule of the form $H \leftarrow B, \alpha(X)$
(2) $\alpha(X)$ is an assumption with $X = vars(B)$
(3) S is a set of atoms including also those for the contrary $c\_\alpha(X)$ of $\alpha(X)$ */
20 $\mathcal{R} := \mathcal{R} \cup \{\rho_d\}$;
21 $\mathcal{A} := \mathcal{A} \cup \{\alpha(X)\}$;
22 $\overline{\alpha(X)} := c\_\alpha(X)$;
/* - Rote learning - */
23 **foreach** $\texttt{c\_}\alpha\texttt{(t)} \in S$ **do**
24 $\mathcal{R}_l := \mathcal{R}_l \cup \{c\_\alpha(X) \leftarrow X = t\}$;
25 **end**
26 **end**
27 **end**
28 **end**
29 **Function** *applyFolding*$(\rho,\mathcal{R})$
30 **while** $foldable(\rho,\mathcal{R})$ **do**
31 $\rho := fold(\rho,\mathcal{R})$;
32 **end**
33 return ρ;
34 **Function** *applyAsmIntro*$(H \leftarrow B,\mathcal{R})$
35 $X := vars(B)$;
36 **if** *there exists* $\alpha(X) \in \mathcal{A}$ *relative to* B **then**
37 $\rho := H \leftarrow B, \alpha(X)$; $S := \emptyset$
38 **if** $\neg sat(ASP(\mathcal{R} \cup \{\rho\},\mathcal{A},\overline{}\rangle,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\emptyset))$ **then**
39 fail;
40 **end**
41 **else** /* introduce an assumption $\alpha(X)$, with a new predicate α */
42 $\rho := H \leftarrow B, \alpha(X)$;
43 $F := \langle\mathcal{R} \cup \{\rho\},\mathcal{A} \cup \{\alpha(X)\},\overline{} \cup \{\alpha(X) \mapsto c\_\alpha(X)\}\rangle$;
44 $S := getAS(ASP(F,\langle\mathcal{E}^+,\mathcal{E}^-\rangle,\{c\_\alpha\}))$;
45 **end**
46 return $\langle\rho,\alpha(X),S\rangle$;

#minimize$\{1, X : p'(X)\}$ directive, the answer set S computed by function *getAS*(P) at line 8 will contain a *minimal* set of new atoms that can be learnt (see lines 9–11) so to obtain a (non-intensional) solution. This directive will also minimize the set of alternative answer sets that are computed at line 8 in case of backtracking.

At line 14 of the *Gen* procedure, the algorithm considers any non-intensional rule $\rho \in \mathcal{R}_l$ of the form $p(X) \leftarrow X = t$. At line 16 the algorithm applies Fact Subsumption to ρ, that is, it checks whether or not it can be deleted by preserving the brave entailment of $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$. If this is not the case, at line 17, it applies the function *applyFolding*, which is defined in a way that (see Definition 3), given a non-intensional rule ρ, it returns an intensional rule ρ_f obtained by applying once or more times (possibly, in a nondeterministic fashion) the Folding transformation using rules in $\mathcal{R}_l \setminus \{\rho\}$.

The applications of the Folding transformation may derive an ABA framework that is no longer a solution of the given learning problem, because the examples may no longer be entailed. Indeed, at line 18, the algorithm checks if the current ABA framework $\langle \mathcal{R} \cup \mathcal{R}_l \cup \{\rho_f\}, \mathcal{A}, \overline{} \rangle$ bravely entails $\langle \mathcal{E}^+, \mathcal{E}^- \rangle$, that is, by Theorem 1, if $ASP(\langle \mathcal{R} \cup \mathcal{R}_l \cup \{\rho_f\}, \mathcal{A}, \overline{} \rangle, \langle \mathcal{E}^+, \mathcal{E}^- \rangle, \emptyset)$ is satisfiable. If it is not, by the following lines 19–25, the ABA framework is transformed into a (possibly non-intensional) solution.

The first step to get again an ABA framework that is a solution of the input learning problem is to apply Assumption Introduction. This is done at line 19, using the function defined at lines 34–46, where the algorithm may either (lines 36–39) take an $\alpha(X)$ in the set $\mathcal{A}$ or (lines 41–44) introduce a new assumption. This choice is a key point for enforcing the termination of the algorithm, as using an assumption in $\mathcal{A}$ may avoid the introduction of an unbounded number of new predicates.

In the case where the algorithm uses an assumption $\alpha(X)$ already belonging to $\mathcal{A}$ (see line 36 and Definition 4) and it does not obtain a solution, then it gets a failure (see line 39) and backtracks to the most recent choice point. This point can be line 19, if *applyAsmIntro* is nondeterministic (that is, the choice of the assumption $\alpha(X) \in \mathcal{A}$ at line 36 is nondeterministic), or line 17, if *applyFolding* is nondeterministic. In the case where no alternative choice is possible at lines 17 and 19, the algorithm halts with failure.

If at line 41 the algorithm introduces a new assumption, then at line 22, it updates the set of assumptions and their contraries. In the case where a new α is introduced, Proposition 2 guarantees that the answer set S to be computed at line 44 exists and, by adding the facts for its contrary c_α via Rote Learning (see lines 23–25), we always get a non-intensional solution for the given brave ABA Learning problem.

We do not provide a concrete definition of the function *applyFolding*, but we require that it satisfies the conditions specified by the following definition, where $bd(\rho)$ denotes the body of rule ρ.

Definition 3. *Let applyFolding$(\rho, \mathcal{R})$ be defined as in Algorithm 1 with two subsidiary functions: (i) foldable$(\rho, \mathcal{R})$, a boolean-valued function such that foldable$(\rho, \mathcal{R})$ implies that $R2$ can be applied to ρ using a rule in $\mathcal{R}$, and (ii) fold$(\rho, \mathcal{R})$, such that fold$(\rho, \mathcal{R})$ is obtained by applying $R2$ to ρ (possibly, in a nondeterministic way). For a sequence $\rho_0, \rho_1, \ldots$ of rules such that $\rho_{i+1} =$ fold$(\rho_i, \mathcal{R})$, for $i \geq 0$, we define a sequence of sets of atom:*

$\mathbb{B}_0 = bd(\rho_0)$

$\mathbb{B}_{i+1} = \mathbb{B}_i \cup \{Eqs_2, K\}$

where B_1, Eqs_1, Eqs_2, K are as in $R2$ and $\{B_1, Eqs_1\} \subseteq \mathbb{B}_i$. We say that applyFolding is bounded *if the following conditions hold:*

F1. if ρ is non-intensional, then foldable$(\rho, \mathcal{R})$ is true;

F2. for all $i \geq 0$, if foldable$(\rho_i, \mathcal{R})$, then (i) $K \notin \mathbb{B}_i$, and (ii) for any variables X, Y, and constant a, if $X = a \in Eqs_2$, then $Y = a \notin \mathbb{B}_i$.

A simple definition of *foldable*$(\rho, \mathcal{R})$ is, for instance, a function which holds true for any ρ that has an occurrence of an equality $X = a$ in its body, where a is a constant, and for every constant c in $\mathcal{L}$, there is a rule $dom(X) \leftarrow X = c$ in $\mathcal{R}$. This function will enforce the termination of *applyFolding* when no constants occur in ρ.

Conditions $F1, F2$ ensure that each application of *applyFolding* terminates, and its output is an intensional rule.

Proposition 3. *Suppose that applyFolding$(\rho, \mathcal{R})$ satisfies the conditions of Definition 3. Then, applyFolding$(\rho, \mathcal{R})$ terminates and returns an intensional rule.*

The language of the ABA framework can be extended by introducing new assumptions and their contraries, and an unbounded introduction of new predicates is a possible source of nontermination of Algorithm 1. The following definition of an *assumption relative to* a rule body will be used to enforce the introduction of a bounded set of assumptions and contraries.

Definition 4. *Suppose that $\rho = H \leftarrow B, \alpha(X)$ is a rule in $\mathcal{R}$ where $\alpha(X) \in \mathcal{A}$. Then we say that $\alpha(X)$ is an assumption relative to B.*

We say that Algorithm *ASP-ABAlearn$_B$ terminates with success* for a given brave ABA Learning problem if it halts and returns a solution. If it halts and does not return a solution, then it *terminates with failure*. Putting together the results proved above, we get the soundness of Algorithm 1.

Theorem 3 (Soundness)**.** *If Algorithm ASP-ABAlearn$_B$ terminates with success for an input brave ABA Learning problem, then its output is an intensional solution.*

Now, by Proposition 3 and the fact that, at line 36 of Algorithm 1, *applyAsmIntro* uses an assumption in the current set $\mathcal{A}$, whenever in $\mathcal{A}$ there exists an assumption relative to the body of the rule under consideration, we get the termination of *ASP-ABAlearn$_B$*.

Theorem 4 (Termination)**.** *Suppose that applyFolding is bounded (see Definition 3). Then Algorithm ASP-ABAlearn$_B$ terminates (either with success or with failure).*

Example 10. *The applications of the transformation rules shown in previous examples for the brave ABA learning problem of Example 3 can be seen as applications of Algorithm 1. Indeed, Example 5 shows an application of RoLe (lines 3–12), Example 6 shows two applications (in different iterations) of applyFolding (line 17) to rules obtained by RoLe, and Example 8 shows an application of applyAsmIntro (line 34) and the subsequent Rote Learning of the facts computed by ASP for the contrary $c\_\alpha(X)$ (lines 23–25). The learning algorithm continues from rule ρ_{17} of Example 8 by performing a new iteration of Procedure Gen. Function applyFolding (line 17) gets:*

ρ_{18}: $c\_\alpha(X) \leftarrow quaker(X)$

The satisfiability test (line 18) fails, as the new ABA framework is not a solution. Now, the applyAsmIntro function proceeds by looking for an assumption relative to $quaker(X)$ in the current set of assumptions (line 36). This assumption is $normal\_quaker(X)$ and, indeed, by replacing ρ_{18} with

ρ_{19}: $c\_\alpha(X) \leftarrow quaker(X), normal\_quaker(X)$

we get an ABA framework that is an intensional solution of the given learning problem (this solution coincides with the ABA framework *including rules $\mathcal{R}_2'$ in Example 3). The learnt ABA framework has (among others) a stable extension including the arguments:*

$\emptyset \vdash quaker(a)$, $\{normal\_quaker(a)\} \vdash pacifist(a)$,
$\{\alpha(b)\} \vdash abnormal\_quaker(b)$,
$\emptyset \vdash democrat(c)$, $\emptyset \vdash pacifist(c)$, $\emptyset \vdash republican(d)$,
$\emptyset \vdash quaker(e)$, $\{normal\_quaker(e)\} \vdash pacifist(e)$.

Note that there are other stable extensions of the resulting ABA framework where, however, either a is not pacifist or b is pacifist, and thus cautious reasoning would not work.

Algorithm *ASP-ABAlearn*$_B$ may terminate with failure in the case where *applyAsmIntro* uses an assumption $\alpha(X)$ already in the current set $\mathcal{A}$ (line 36), the resulting rule ρ: $H \leftarrow B, \alpha(X)$ does not produce a solution (line 38), and no alternative application of Folding is available when backtracking to line 17. Algorithm *ASP-ABAlearn*$_B$ may halt with failure even in cases where a solution exists, but computing it would require, for instance, introducing other assumptions. In this sense *ASP-ABAlearn*$_B$ is not complete. However, the learning algorithm can be slightly modified so that it always terminates with success whenever the learning problem has a solution, possibly returning a non-intensional solution. This modification is realised by allowing *applyFolding*$(\rho, \mathcal{R})$ to return ρ after having tried unsuccessfully all possible applications of the Folding transformation. Thus, the resulting rule will be non-intensional. The modified algorithm is called *ASP-ABAlearn*$_{BE}$ (E stands for *Enumerating*).

Theorem 5 (Soundness and Completeness of *ASP-ABAlearn*$_{BE}$)**.** *For all brave ABAlearn problems, ASP-ABAlearn*$_{BE}$ *terminates and returns a, possibly non-intensional, solution, if a solution exists.*

7 Implementation and Experiments

We have realised a proof-of-concept implementation [8] of our *ASP-ABAlearn*$_B$ strategy using the SWI-Prolog system[4] and the Clingo ASP solver[5]. We have used Prolog as a fully fledged programming language to handle symbolically the rules and to implement the nondeterministic search for a solution to the learning problem, while we have used ASP as a specialised solver for computing answer sets corresponding to stable extensions. In particular, our tool consists of two Prolog modules implementing *RoLe* and *Gen* and two further modules implementing (i) the ASP encoding of Definition 2, and (ii) the API to invoke Clingo from SWI-Prolog and collect the answer sets to be used by *RoLe* and *Gen*.

Table 1 reports the results of the experimental evaluation we have conducted on a benchmark set consisting of seven classic learning problems taken from the literature (*flies* [9], *innocent* [2], *nixon_diamond* [24], and variants thereof), and three larger problems (i.e., tabular datasets) from [32], to show that our approach works for non-trivial, non-ad-hoc examples. The discussion on scalability is out of the scope of the present paper.

In the table, we compare *ASP-ABAlearn*$_B$ with ILASP, a state-of-the-art learner for ASP programs[6]. When running ILASP we have opted for adopting the most direct representations of the learning problems, in terms of mode declarations. [7] In the ILASP column, *unsat* indicates that the system halted within the timeout, but was unable to learn an ASP program. These *unsat* results are due to the fact that the predicates and the mode declarations specified in the background knowledge are not sufficient to express a solution for the learning problem. In this class of problems, the use of Assumption Introduction proposed in this paper may demonstrate its advantages. Indeed, for instance, we have also tried ILASP on the *acute* dataset extending the background knowledge with additional information matching the use of assumptions and their contraries automatically introduced by *ASP-ABAlearn*$_B$. This manual addition has allowed ILASP to learn a solution in about 37 seconds.

We refrained from comparing *ASP-ABAlearn*$_B$ with tools like FOLD-RM [28, 32], which can only learn *stratified* normal logic programs. These programs admit a single stable model and brave learning is not significant for them.

Table 1. Experimental results. Experiments have run on an Apple M1 equipped with 8 GB of RAM, setting a *timeout* of 15 minutes. Times are in seconds. Column *ASP-ABAlearn*$_B$ reports the sum of the CPU and System times taken by out tool to compute a solution for the Learning problem, while column ILASP reports the time taken by ILASP system log. Columns *BK*, $\mathcal{E}^+$, and $\mathcal{E}^-$ report the number of rules in the background knowledge, and the number of positive and negative examples, respectively.

Learning problem	*BK*	$\mathcal{E}^+$	$\mathcal{E}^-$	*ASP-ABAlearn*$_B$	ILASP
flies	8	4	2	0.01	0.09
flies_birds&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*

8 Conclusions

We have designed an approach for learning ABA frameworks based on transformation rules [22], and we have shown that, in the case of brave reasoning under the stable extension semantics, many of the reasoning tasks used by that strategy can be implemented through an ASP solver. We have studied a number of properties concerning both the transformation rules and an algorithm that implements our learning strategy, including its termination, soundness, and completeness, under suitable conditions. A distinctive feature of our approach is that argumentation plays a key role not only at the representation level, as we learn defeasible rules represented by ABA frameworks, but also at the meta-reasoning level, as our learning strategy can be seen as a form of debate that proceeds by conjecturing general rules that cover the examples and then finding exceptions to them.

Even if the current implementation is not optimised, it allows solving some non trivial learning problems. The most critical issue is that the application of Folding, needed for generalisation, is non-deterministic, as there may be different choices for the rules to be used for applying it. Currently, we are experimenting various mechanisms to control Folding for making it more deterministic. In addition to refining the implementation, we are also planning to perform a more thorough experimental comparison with non-monotonic ILP systems (such as FOLD-RM [28, 32] and ILASP [16]). Further extensions of our ABA Learning approach can be envisaged, exploiting the ability of ABA frameworks to be instantiated to different logics and semantics, and possibly address the problem of learning non-flat ABA frameworks [2, 4]. To this aim we may need to integrate ABA Learning with tools that go beyond ASP solvers (e.g. [17]).

[4] SWI-Prolog v9.0.4, https://www.swi-prolog.org/
[5] Clingo v5.6.2, https://potassco.org/clingo/
[6] ILASP v4.4.0, using option `--version=2`, https://doc.ilasp.com/
[7] The *ASP-ABAlearn*$_B$ and ILASP specifications are archived in [8].

Acknowledgements

We thank support from the Royal Society, UK (IEC\R2\222045 - International Exchanges 2022). Toni was partially funded by the ERC under the EU's Horizon 2020 research and innovation programme (grant agreement No. 101020934) and by J.P. Morgan and the Royal Academy of Engineering, UK, under the Research Chairs and Senior Research Fellowships scheme. This paper has also obeen partially supported by the Italian MUR PRIN 2022 Project DOMAIN (2022TSYYKJ, CUP B53D23013220006, PNRR M4.C2.1.1) funded by the European Union – NextGenerationEU. De Angelis and Proietti are members of the INdAM-GNCS research group.

References

[1] C. Aravindan and P. Dung. On the correctness of unfold/fold transformation of normal and extended logic programs. *J. Log. Program.*, 24 (3):201–217, 1995. doi: 10.1016/0743-1066(94)00104-E.

[2] A. Bondarenko, P. Dung, R. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artif. Intell.*, 93: 63–101, 1997. doi: 10.1016/S0004-3702(97)00015-5.

[3] O. Cocarascu, A. Stylianou, K. Cyras, and F. Toni. Data-empowered argumentation for dialectically explainable predictions. In *Proc. ECAI 2020*, FAIA 325, pages 2449–2456. IOS Press, 2020. doi: 10.3233/FAIA200377.

[4] K. Cyras, X. Fan, C. Schulz, and F. Toni. Assumption-based argumentation: Disputes, explanations, preferences. *FLAP*, 4(8), 2017. URL http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf.

[5] K. Cyras, T. Oliveira, A. Karamlou, and F. Toni. Assumption-based argumentation with preferences and goals for patient-centric reasoning with interacting clinical guidelines. *Argument Comput.*, 12(2):149–189, 2021. doi: 10.3233/AAC-200523.

[6] E. De Angelis, M. Proietti, and F. Toni. ABA learning via ASP. *EPTCS*, 385:1–8, aug 2023. doi: 10.4204/eptcs.385.1. Technical Communications, ICLP 2023.

[7] E. De Angelis, M. Proietti, and F. Toni. Learning brave assumption-based argumentation frameworks via ASP. arXiv preprint arXiv:2408.10126. http://arxiv.org/abs/2408.10126, 2024. Full version of this paper.

[8] E. De Angelis, M. Proietti, and F. Toni. Code and data for "Learning Brave Assumption-Based Argumentation Frameworks via ASP". Zenodo, 2024. Available at https://doi.org/10.5281/zenodo.13330013.

[9] Y. Dimopoulos and A. C. Kakas. Learning non-monotonic logic programs: Learning exceptions. In *ECML 1995*, pages 122–137, 1995. doi: 10.1007/3-540-59286-5_53.

[10] P. Dung, R. Kowalski, and F. Toni. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*, pages 199–218. Springer, 2009. doi: 10.1007/978-0-387-98197-0_10.

[11] X. Fan. A temporal planning example with assumption-based argumentation. In *PRIMA 2018*, LNCS 11224, pages 362–370. Springer, 2018. doi: 10.1007/978-3-030-03098-8_22.

[12] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice*. Morgan & Claypool Publishers, 2012.

[13] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991. doi: 10.1007/BF03037169.

[14] K. Inoue and H. Haneda. Learning abductive and nonmonotonic logic programs. In *Abduction and Induction: Essays on their Relation and Integration*, pages 213–231. Kluwer Academic, 2000. doi: 10.1007/978-94-017-0606-3_14.

[15] K. Inoue and Y. Kudoh. Learning extended logic programs. In *IJCAI*, pages 176–181. Morgan Kaufmann, 1997.

[16] M. Law, A. Russo, and K. Broda. Inductive learning of answer set programs. In *JELIA 2014*, LNCS 8761, pages 311–325, 2014. doi: 10.1007/978-3-319-11558-0_22.

[17] T. Lehtonen, J. P. Wallner, and M. Järvisalo. Declarative algorithms and complexity results for assumption-based argumentation. *J. Artif. Intell. Res.*, 71:265–318, 2021. doi: 10.1613/JAIR.1.12479.

[18] S. Muggleton. Inverse entailment and Progol. *New generation computing*, 13(3-4):245–286, 1995. doi: 10.1007/BF03037227.

[19] R. P. Otero. Induction of stable models. In C. Rouveirol and M. Sebag, editors, *ILP 2001*, LNCS 2157, pages 193–205. Springer, 2001. doi: 10.1007/3-540-44797-0_16.

[20] A. Pettorossi and M. Proietti. Transformation of logic programs: Foundations and techniques. *J. Log. Program.*, 19/20:261–320, 1994. doi: 10.1016/0743-1066(94)90028-0.

[21] N. Prentzas, A. Gavrielidou, M. Neofytou, and A. C. Kakas. Argumentation-based explainable machine learning (ArgEML): A real-life use case on gynecological cancer. In *ArgML 2022*, CEUR-WS, 3208, pages 1–13, 2022. URL https://ceur-ws.org/Vol-3208/paper1.pdf.

[22] M. Proietti and F. Toni. Learning assumption-based argumentation frameworks. In *ILP 2022*, Lecture Notes in Computer Science (LNAI 13779), pages 100–116. Springer, 2024. doi: 10.1007/978-3-031-55630-2_8.

[23] O. Ray. Nonmonotonic abductive inductive learning. *J. Appl. Log.*, 7 (3):329–340, 2009. doi: 10.1016/j.jal.2008.10.007.

[24] R. Reiter and G. Criscuolo. On interacting defaults. In *IJCAI*, pages 270–276. William Kaufmann, 1981. URL http://ijcai.org/Proceedings/81-1/Papers/054.pdf.

[25] C. Sakama. Inverse entailment in nonmonotonic logic programs. In *Proc. ILP 2000*, 2000. doi: 10.1007/3-540-44960-4_13.

[26] C. Sakama. Induction from answer sets in nonmonotonic logic programs. *ACM TOCL*, 6(2):203–231, 2005. doi: 10.1145/1055686.1055687.

[27] C. Sakama and K. Inoue. Brave induction: A logical framework for learning from incomplete information. *Mach. Learn.*, 76(1):3–35, 2009. doi: 10.1007/s10994-009-5113-y.

[28] F. Shakerin, E. Salazar, and G. Gupta. A new algorithm to automate inductive learning of default theories. *Theory and Practice of Logic Programming*, 17(5-6):1010–1026, 2017. doi: 10.1017/S1471068417000333.

[29] C. G. Tirsi, M. Proietti, and F. Toni. ABALearn: An automated logic-based learning system for ABA frameworks. In *AIxIA 2023*, LNCS 14318, pages 3–16. Springer, 2023. doi: 10.1007/978-3-031-47546-7_1.

[30] F. Toni. A tutorial on assumption-based argumentation. *Argument & Computation*, 5(1):89–117, 2014. doi: 10.1080/19462166.2013.869878.

[31] F. Toni and R. Kowalski. An argumentation-theoretic approach to logic program transformation. In *LOPSTR 1995*, LNCS 1048, pages 61–75, 1996. doi: 10.1007/3-540-60939-3_5.

[32] H. Wang, F. Shakerin, and G. Gupta. FOLD-RM: A scalable, efficient, and explainable inductive learning algorithm for multi-category classification of mixed data. *Theory and Practice of Logic Programming*, 22 (5):658–677, 2022. doi: 10.1017/S1471068422000205.