Embedding Defeasible Argumentation in Answer Set Programming

Peter Novák
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
peter.novak@tu-clausthal.de

Martin Baláž
Department of Applied Informatics
Comenius University
Bratislava, Slovakia
balaz@ii.fmph.uniba.sk

Jürgen Dix
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
dix@tu-clausthal.de

Abstract

We investigate the relationship between the framework of *Defeasible Logic Programming* (DeLP) and *Answer Set Programming* (ASP). Firstly, we give a characterization of *inadmissible sets* of an argumentation framework in terms of relations between its preferred extensions, resp. admissible sets. Secondly, we use this result to embed *Defeasible Logic Programming* in *Answer Set Programming*. We provide two translations of DeLP programs into extended logic programs. Thirdly, we provide generic algorithms for extracting the set of warranted literals of the original defeasible logic program from answer sets of the translated program.

Introduction

Defeasible Logic Programming (DeLP) (García & Simari 2004) is a logic programming framework for logical argumentative reasoning (Besnard & Hunter 2000) based on defeasible argumentation (Prakken & Vreeswijk 2002). In DeLP, an argumentation formalism is used for deciding whether a query is true w.r.t. a defeasible logic program (*delp*). Queries are supported by arguments which can be defeated by other arguments. A query is true only when the argument supporting it is a warrant, i.e. the DeLP dialectical analysis warrant procedure found it to be undefeated.

According to (Thimm & Kern-Isberner 2008), there's only little reported on the relationship of DeLP to other defeasible and non-monotonic reasoning frameworks. In this paper we investigate the relationship of DeLP to *Answer Set Programming* (ASP) (for a comprehensive discussion consult (Baral 2003)), a non-monotonic reasoning framework based on stable model semantics (Gelfond & Lifschitz 1988) for logic programs.

The first result on relating DeLP and ASP in (Thimm & Kern-Isberner 2008), establishes two translations from defeasible logic programs with an empty preference relation to extended logic programs. It exploits the notion of *minimal disagreement sets*, minimal contradictory sets of literals of a logic program w.r.t. the strict rules of a *delp*. The established link between the two programs, however, is quite weak as Thimm and Kern-Isberner could only prove that (1) each warranted literal is contained in at least one answer set of the resulting extended logic program (for a straightforward translation), and (2) the set of warranted literals is a subset of the intersection of the answer sets.

With the same motivation as Thimm and Kern-Isberner, in this paper we investigate the relationship of DeLP with an empty preference relation to ASP and show how DeLP can be embedded in the ASP framework. We introduce two translations of defeasible logic programs into extended logic programs. By analysing their properties, we characterize also the whole class of such translations. We show, that from answers sets of the resulting logic programs we can extract the set of warranted literals.

In the remainder of this paper, we first introduce Dung's (Dung 1995) theory of argumentation frameworks and conclude the section by introducing some original theoretical results regarding properties of inadmissible sets of arguments. Then we characterize Defeasible Logic Programming as an *instantiation* of an abstract argumentation framework and introduce the framework of Answer Set Programming. The main contribution of this paper, however, is the following embedding of DeLP in ASP based on the theoretical results for abstract argumentation frameworks. We conclude the paper by a brief discussion of our results and a comparison to related work.

Argumentation frameworks

In accord with Dung's abstract argumentation framework (Dung 1995), for now we keep the notion of argument to be an abstract entity whose role is solely determined by its relations to other arguments.

In the following, we reiterate a slightly less abstract version of the set of Dung's definitions. The introductory definitions are adapted from (Dung 1995).

Definition 1 (argument structure) *An* argument $\mathcal{A}$ *is a set of interrelated pieces of knowledge supporting a claim* h *from evidence. We also say that* $\mathcal{A}$ *is a* support *for the* conclusion h. *A tuple* $\langle \mathcal{A}, h \rangle$ *is called an* argument structure.

When the context will be clear, in the following we will use the notions of *an argument structure* and *an argument* interchangeably. I.e. if S is a set of argument structures and $\langle \mathcal{A}, h \rangle \in S$, we will also simply say that $\mathcal{A}$ is an argument from S ($\mathcal{A} \in S$) and w.l.o.g. we will use the relations defined over argument structures for arguments as well.

Definition 2 (argumentation framework) *An* argumentation framework *is a tuple $AF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$, where $\mathbb{U}$ is a set of argument structures, $\mathbf{R} \subseteq \mathbb{U} \times \mathbb{U}$ is an* attack *relation between argument structures and $\sqsubseteq$ denotes a* subargument relation*, i.e. a partial order over U. Additionally each argument structure $\langle \mathcal{A}, h \rangle \in \mathbb{U}$ satisfies the following conditions:*

self-consistency: *$\mathcal{A}$ is* self-consistent *w.r.t. R iff there are no $\mathcal{A}', \mathcal{A}'' \sqsubseteq \mathcal{A}$ s.t. $\mathcal{A}'\mathbf{R}\mathcal{A}''$, nor $\mathcal{A}''\mathbf{R}\mathcal{A}'$; and*

minimality: *$\mathcal{A}$ is* minimal *iff $\mathcal{A}$ supports h and there is no $\mathcal{A}' \in \mathbb{U}$ s.t. $\mathcal{A}' \sqsubseteq \mathcal{A}$ and $\mathcal{A}'$ supports h.*

If $\mathcal{A}' \sqsubseteq \mathcal{A}$, we say that $\mathcal{A}'$ is a subargument *of $\mathcal{A}$ and $\mathcal{A}$ is a* superargument *of $\mathcal{A}'$. An argument $\mathcal{A}'$, such that $\mathcal{A}'\mathbf{R}\mathcal{A}$ is called a* defeater *of $\mathcal{A}$. If there is no defeater of an argument $\mathcal{A}$ in $\mathbb{U}$, i.e. $\forall \mathcal{B} \in \mathbb{U} : (\mathcal{A}, \mathcal{B}) \notin \mathbf{R}$, the argument $\mathcal{A}$ is said to be* undisputed.

For brevity, we use also notation $\mathcal{A} \in \mathbb{U}$ with $\mathcal{A}$ being an argument s.t. there exists an argument structure $\langle \mathcal{A}, h \rangle \in \mathbb{U}$. Similarly for sets of arguments we will use $S \subseteq \mathbb{U}$.

In the following we will take a closer look at properties of sets of arguments.

Definition 3 (properties of argument sets) *Let $\langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$ be an argumentation framework and $S \subseteq \mathbb{U}$ be a set of argument structures.*

We say that S attacks *an argument $\mathcal{A}$ if $\mathcal{A}$ is attacked by some argument from S, i.e. $\exists \mathcal{B} \in S : \mathcal{B}\mathbf{R}\mathcal{A}$.*

S is conflict-free *iff there are no arguments $\mathcal{A}$ and $\mathcal{B}$ from S, s.t. $\mathcal{A}$ attacks $\mathcal{B}$.*

We say that an argument $\mathcal{A} \in \mathbb{U}$ is acceptable *w.r.t. S iff for each $\mathcal{B} \in \mathbb{U}$ holds: If $\mathcal{B}\mathbf{R}\mathcal{A}$, then $\mathcal{B}$ is attacked by S.*

Finally, a conflict-free set of arguments S is admissible *iff each argument from S is acceptable w.r.t. S.*

Definition 4 (preferred extension) *A* preferred extension *of an argumentation framework AF is a maximal (w.r.t. inclusion) admissible set of argument structures.*

A conflict-free set of arguments S is called a stable extension *of AF iff S attacks each argument $\mathcal{A}$ which does not belong to S.*

Below, we introduce a useful property of admissible sets w.r.t. preferred extensions of argumentation frameworks.

Proposition 1 *Let AF be an argumentation framework.*

For each admissible set S of AF, there exists a preferred extension E of AF such that $S \subseteq E$.

Proof. For a proof see Theorem 1 in (Dung 1995). □

The Proposition 1 holds also for singleton argument sets which are trivially admissible:

Corollary 1 *For each argument structure $\langle \mathcal{A}, h \rangle \in \mathbb{U}$ there is a preferred extension E of AF, s.t. $\langle \mathcal{A}, h \rangle \in E$.*

The following Definition 5, Theorem 1 and Corollaries 2 and 3 are the main theoretical results concluding this section.

Definition 5 (inadmissible set) *Let $AF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$ be an argumentation framework and $S \subseteq \mathbb{U}$ be a set of arguments. Let also $\mathcal{A} \in \mathbb{U}$ be an argument from AF.*

*S is said to be $\mathcal{A}$-*unacceptable *iff $S \cup \mathcal{A}$ is not conflict free, i.e. there exists $\mathcal{B} \in S$, s.t. either $\mathcal{A}\mathbf{R}\mathcal{B}$, or $\mathcal{B}\mathbf{R}\mathcal{A}$. A maximal $\mathcal{A}$-unacceptable set S is said to be $\mathcal{A}$-*inadmissible.

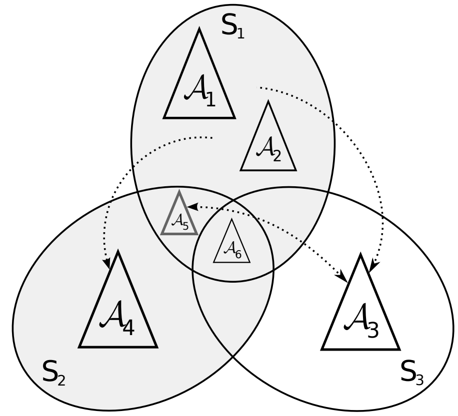

Figure 1: Example of relationships between preferred extensions of an argumentation framework.

The following theorem establishes an alternative characterization of an inadmissible set.

Theorem 1 *Let $AF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$ be an argumentation framework with an irreflexive attack relation $\mathbf{R}$. Let also $\mathcal{A}$ be an argument in AF and $\mathcal{S}^*$ be the set of all preferred extensions of AF. Let $\overline{\mathcal{S}}^{\mathcal{A}}$ be a set of arguments defined as follows*

$$\overline{\mathcal{S}}^{\mathcal{A}} = \mathbb{U} \setminus (\bigcup_{\substack{S \in \mathcal{S}^* \\ \mathcal{A} \in S}} S)$$

.

The set $\overline{\mathcal{S}}^{\mathcal{A}}$ is an $\mathcal{A}$-inadmissible set.

Proof. Corollary 1 implies $\mathbb{U} = \bigcup_{S \in \mathcal{S}^*} S$. We assume $\overline{\mathcal{S}}^{\mathcal{A}}$ is not $\mathcal{A}$-inadmissible, i.e. it contains an argument $\mathcal{B} \in \overline{\mathcal{S}}^{\mathcal{A}}$, s.t. either $\neg(\mathcal{B}\mathbf{R}\mathcal{A})$ and $\neg(\mathcal{A}\mathbf{R}\mathcal{B})$. From that we have, that $\{\mathcal{A}, \mathcal{B}\}$ is admissible. Proposition 1 says that there must be a preferred extension E, s.t. $\mathcal{A}, \mathcal{B} \in E$ and $E \in \mathcal{S}^*$. By necessity we have $\mathcal{B} \in E \subseteq \bigcup_{\substack{S \in \mathcal{S}^* \\ \mathcal{A} \in S}} S$, hence $\mathcal{B} \notin \overline{\mathcal{S}}^{\mathcal{A}}$, which contradicts the definition of $\mathcal{B}$. □

Corollary 2 *Let $\mathcal{A}$ be an argument in AF, and $\mathcal{S}$ be a set of admissible sets of AF containing all the preferred extensions of AF. Then it also holds $\overline{\mathcal{S}}^{\mathcal{A}} = \mathbb{U} \setminus (\bigcup_{\substack{S \in \mathcal{S} \\ \mathcal{A} \in S}} S)$.*

Proof. Follows directly from Theorem 1 and the fact that for sets A, B, C, s.t. $B \subset C$ holds $A \setminus C = A \setminus (B \cup C)$. □

Here is another useful consequence of Theorem 1:

Corollary 3 *Let AF be an argumentation framework as in Theorem 1 and $\mathcal{S}^*$ be the set of its preferred extensions.*

For each undisputed argument $\mathcal{A}$ of AF: $\mathcal{A} \in \bigcap_{S \in \mathcal{S}^*} S$.

Proof. We use Theorem 1 and Corollary 1. Let S be a preferred extension $S \in \mathcal{S}^*$, s.t. $\mathcal{A} \notin S$ and S attacks $\mathcal{A}$. Then there must be an argument $\mathcal{B} \in S$, s.t. $\mathcal{A}\mathbf{R}\mathcal{B}$, which contradicts the fact that $\mathcal{A}$ is undisputed. □

Figure 1 shows an example of relationships between preferred extensions of an argumentation framework and application of Theorem 1. Dotted arrows depict attacking relation between arguments. Note, that for each argument, they point only to sets to which that argument does not belong. Considering the argument $\mathcal{A}_5$, the grey area depicts the union of all preferred extensions $\mathcal{A}_5$ belongs to. The only argument outside of this area, $\mathcal{A}_3$, is the only defeater of $\mathcal{A}_5$. Also note, that $\mathcal{A}_6$ is undisputed since it belongs to the intersection of all the preferred extensions of AF.

The theoretical results introduced in this section will be used later to embed the framework of Defeasible Logic Programming into Answer Set Programming.

Defeasible Logic Programming

Defeasible Logic Programming (García & Simari 2004) is a logic programming language allowing modelling of defeasible knowledge. It can be seen as an instance of Dung's abstract argumentation framework introduced above. In order to instantiate DeLP as an argumentation framework $DeLPAF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$ scheme, we need to provide a definition of a valid argument structure (thus defining $\mathbb{U}$), the subargument relation $\sqsubseteq$ and the DeLP attack relation $\mathbf{R}$.

The following subsections first provide a definition for DeLP arguments and the subargument relation and subsequently we introduce a specification of the attack relation. The main point of this section is a formal introduction of DeLP as an instance of the abstract argumentation framework and introducing the notion of a *warranted literal*.

The formalism introduced below is adapted from (Thimm & Kern-Isberner 2008).

Arguments

A defeasible logic program (*delp*) consists of a set of rules and is divided into two parts: *Strict* knowledge and *defeasible* knowledge. Strict rules are meant to derive certain knowledge, while defeasible rules derive uncertain, or defeasible knowledge.

For the purposes of this paper we define DeLP over a first-order language without function symbols except constants.

Definition 6 (literals) *A literal L is either an atom A, or a (classical) negated atom $\neg A$. $\mathfrak{Lit}$ denotes a set of all* literals.

Definition 7 (facts and rules) *A* fact *is a literal $L \in \mathfrak{Lit}$.*

Let $L \in \mathfrak{Lit}$ be a literal and $B \subseteq \mathfrak{Lit}$ be a non-empty set of literals. A strict rule *is an ordered pair of the form $L \leftarrow B$ and a* defeasible rule *is an ordered pair $L \prec B$. By $head(r)$ we denote the literal L of the rule r and $body(r) = B$ denotes the set of the body literals.*

Definition 8 (*delp*) *A* defeasible logic program *(delp) $\mathcal{P} = (\Pi, \Delta)$ consists of a set Π of facts and strict rules and a set Δ of defeasible rules (both possibly infinite).*

Given a *delp*, we derive literals as follows.

Definition 9 (defeasible derivation) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp *and let $h \in \mathcal{L}$ be a literal. A* (defeasible) derivation *of h from $\mathcal{P}$, denoted $\mathcal{P} \mid\!\sim h$, consists of a finite sequence $h_1, \ldots, h_n = h$ of literals ($h_i \in \mathfrak{Lit}$), s.t. h_i is a fact ($h_i \in \Pi$), or there exists a strict or defeasible rule $r \in \mathcal{P}$, s.t. $head(r) = h_i$ and $body(r) = b_1, \ldots, b_k$, where every b_l ($1 \leq l \leq k$) is an element h_j with $j < i$.*

We also say that a program $\mathcal{P} = (\Pi, \Delta)$ is *contradictory* when $\Pi \cup \Delta \mid\!\sim \bot$, with usual denotation of $\bot = p \wedge \neg p$ for some $p \in \mathfrak{Lit}$. In the following we assume only consistent sets of strict rules, i.e. $\Pi \not\mid\!\sim \bot$. For a detailed discussion on ramifications of this restriction see (García & Simari 2004), Observation 2.3.

Definition 10 (argument, subargument) *Let $h \in \mathfrak{Lit}$ be a literal and let $\mathcal{P} = (\Pi, \Delta)$ be a* delp.

$\mathcal{A}$ is an argument for h, iff 1) $\mathcal{A} \subseteq \Delta$, 2) there exists a defeasible derivation of h from $\mathcal{P}' = (\Pi, \mathcal{A})$, 3) the set $\Pi \cup \mathcal{A}$ is non-contradictory, and 4) $\mathcal{A}$ is minimal w.r.t. inclusion.

$\langle \mathcal{A}, h \rangle$ denotes an argument structure *from $\mathcal{P}$. Furthermore, we say that an argument $\mathcal{B}$, is a* subargument *of $\mathcal{A}$ iff $\mathcal{B} \subseteq \mathcal{A}$. In turn $\langle \mathcal{B}, h' \rangle \sqsubseteq \langle \mathcal{A}, h \rangle$, when $\mathcal{B}$ is an argument for h'.*

Attack relation

To finally instantiate DeLP as an argumentation framework, it remains to specify the relation of attacking.

Definition 11 (disagreement and counterargument) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp. *Two literals h and h'* disagree *iff the program $\Pi \cup \{h, h'\}$ is contradictory.*

$\langle \mathcal{A}_1, h_1 \rangle$ is a counterargument structure *to an argument structure $\langle \mathcal{A}_2, h_2 \rangle$ (both from $\mathcal{P}$) at a literal $h \in \mathfrak{Lit}$, iff there exists a subargument structure $\langle \mathcal{A}, h \rangle \sqsubseteq \langle \mathcal{A}_2, h_2 \rangle$, s.t. h and h_1 disagree. Additionally, when $h = h_2$, then $\langle \mathcal{A}_1, h_1 \rangle$ is a* direct attack *on $\langle \mathcal{A}_2, h_2 \rangle$, and* indirect attack *otherwise.*

The DeLP attack relation is based on preferences between arguments.

Definition 12 (preference criterion $\prec$) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp *and $\mathbb{U}$ be a set of all argument structures from $\mathcal{P}$. A preference criterion among arguments is an irreflexive and antisymetric relation $\prec \subseteq \mathbb{U} \times \mathbb{U}$. If $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ are argument structures from $\mathcal{P}$, $\langle \mathcal{A}_1, h_1 \rangle$ will be* strictly preferred *over $\langle \mathcal{A}_2, h_2 \rangle$ iff $\langle \mathcal{A}_2, h_2 \rangle \prec \langle \mathcal{A}_1, h_1 \rangle$.*

Definition 13 (defeater) *An argument structure $\langle \mathcal{A}_1, h_1 \rangle$ is a* defeater *of an argument structure $\langle \mathcal{A}_2, h_2 \rangle$ iff there is a subargument $\langle \mathcal{A}, h \rangle \sqsubseteq \langle \mathcal{A}_2, h_2 \rangle$, s.t. $\langle \mathcal{A}_1, h_1 \rangle$ is a counterargument structure of $\langle \mathcal{A}_2, h_2 \rangle$ at literal h and either $\langle \mathcal{A}, h \rangle \prec \langle \mathcal{A}_1, h_1 \rangle$* (proper defeat*), or $\langle \mathcal{A}, h \rangle \not\prec \langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_1, h_1 \rangle \not\prec \langle \mathcal{A}, h \rangle$* (blocking defeat*).*

The notion of an argument defeater provides the DeLP attack relation between arguments. Finally, we can conclude instantiate DeLP as an argumentation framework w.r.t. Dung's theoretical foundation.

Definition 14 (DeLP argumentation framework) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp *and $\mathbb{U}$ be the set of argument structures from $\mathcal{P}$.* Defeasible Logic Programming argumentation framework *of $\mathcal{P}$ is an argumentation framework $DeLPAF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq, \prec \rangle$ extended with the preference criterion $\prec$ over $\mathbb{U}$. The subargument relation $\sqsubseteq \subseteq \mathbb{U} \times \mathbb{U}$ is* *the subargument relation as defined in Definition 10. Finally the relation of attacking* $\mathbf{R}$ *is defined as* $\langle \mathcal{A}_1, h_1 \rangle \mathbf{R} \langle \mathcal{A}_2, h_2 \rangle$ *iff* $\langle \mathcal{A}_1, h_1 \rangle$ *is a defeater of* $\langle \mathcal{A}_2, h_2 \rangle$.

Warrants

The semantics of *delp* program $\mathcal{P}$ as a query evaluation system is formally defined in terms of warranted literals of $\mathcal{P}$. The definitions below introduce a procedure for extracting warranted literals from a *delp*.

Definition 15 (argumentation line) *Let* $DeLPAF$ *be a DeLP argumentation framework of a* delp $\mathcal{P} = (\Pi, \Delta)$.

An argumentation line λ *in* $DeLPAF$ *is any finite sequence of argument structures* $[\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \ldots]$ *s.t.* $\mathcal{A}_i \mathbf{R} \mathcal{A}_{i+1}$ *for each* $1 < i \leq n$.

λ *is said to be* acceptable *iff*

1. λ *is finite,*
2. *every argument structure* $\langle \mathcal{A}_i, h_i \rangle$ *with* $i > 0$ *is a defeater of its predecessor* $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ *and if* $\langle \mathcal{A}_i, h_i \rangle$ *is a blocking defeater of* $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$ *and* $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ *exists, then* $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ *is a proper defeater of* $\langle \mathcal{A}_i, h_i \rangle$,
3. *both* $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_3 \cup \ldots$ *and* $\Pi \cup \mathcal{A}_2 \cup \mathcal{A}_4 \cup \ldots$ *are non-contradictory (*concordance *of supporting and interfering arguments respectively), and*
4. *no argument structure* $\langle \mathcal{A}_k, h_k \rangle$ *is a subargument of* $\langle \mathcal{A}_i, h_i \rangle$ *with* $i < k$.

The conditions on acceptability of an argumentation line instantiates a notion of a dialectical constraint.

From sets of acceptable argumentation lines with the same argumentation structure in the first position, we construct dialectical trees, the basis for deciding whether the literal supported by the root is warranted, or not.

Definition 16 (dialectical tree) *Let* $\langle \mathcal{A}_0, h_0 \rangle$ *be an argument structure of a DeLP argumentation framework* $DeLPAF$ *of a* delp $\mathcal{P} = (\Pi, \Delta)$. *A dialectical tree for* $\langle \mathcal{A}_0, h_0 \rangle$, *denoted* $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, *is defined as follows:*

1. *The root of the tree is* $\langle \mathcal{A}_0, h_0 \rangle$,
2. *let* $\langle \mathcal{A}_n, h_n \rangle$ *be a node in* $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$ *and let* $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ *be the sequence of nodes from the root to* $\langle \mathcal{A}_n, h_n \rangle$. *Let* $\langle \mathcal{B}_1, q_1 \rangle, \ldots, \langle \mathcal{B}_k, q_k \rangle$ *be the defeaters of* $\langle \mathcal{A}_n, h_n \rangle$. *For every defeater* $\langle \mathcal{B}_i, q_i \rangle$ *with* $1 \leq i \leq k$, *s.t. the argumentation line* $\lambda' = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}_i, q_i \rangle]$ *is acceptable, the node* $\langle \mathcal{A}_n, h_n \rangle$ *has a child* $\langle \mathcal{B}_i, q_i \rangle$. *If there is no such* $\langle \mathcal{B}_i, q_i \rangle$, *then the node* $\langle \mathcal{A}_n, h_n \rangle$ *is a leaf.*

The following marking criterion articulates relations between argument structures in a dialectical tree.

Definition 17 (marking criterion) *Let* $DeLPAF$ *be a DeLP argumentation framework of a* delp $\mathcal{P} = (\Pi, \Delta)$. *Let also* $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ *be a dialectical tree for* $\langle \mathcal{A}, h \rangle$ *in* $DeLPAF$. *The* marked dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ *is defined as follows*

1. *every leaf of* $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$ *will be marked* U *in* $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$,
2. *let* $\langle \mathcal{B}, q \rangle$ *be an inner node of* $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$. *The node* $\langle \mathcal{B}, q \rangle$ *will be marked* U *in* $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ *iff every child of* $\langle \mathcal{B}, q \rangle$ *is marked* D *in* $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$. *The node* $\langle \mathcal{B}, q \rangle$ *will be marked* D, *iff at least one of its children is marked* U *in* $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$.

Finally, depending on the marking of the root node, the DeLP query is decided.

Definition 18 (warrant) *Let* $\mathcal{P}$ *be a* delp *and* $DeLPAF$ *be a DeLP argumentation framework of* $\mathcal{P}$. *A literal* $h \in \mathfrak{Lit}$ *is* warranted *in* $\mathcal{P}$, *iff there exists an argument structure* $\langle \mathcal{A}, h \rangle$ *in* $DeLPAF$, *s.t. the root of the marked dialectical tree* $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ *is marked* U, *i.e. the argument* $\mathcal{A}$ *is undefeated. We say that* h *is a* warranted literal *and* $\mathcal{A}$ *is a* warrant *for* h.

Answer Set Programming

This section provides a brief overview of of *Answer Set Programming* framework as proposed by Gelfond and Lifschitz in (Gelfond & Lifschitz 1988). Similarly to (Thimm & Kern-Isberner 2008), we consider here extended logic programs, which distinguish between classical $\neg$ and default negation $\sim$.

To provide the syntax of the ASP framework, we use the same set of literals $\mathfrak{Lit}$ as defined above. Additionally we assume $\neg\neg L = L$ for $L \in \mathfrak{Lit}$. In the following we use the same functions *head/1* and *body/1* as defined for *delp* for denoting the head and the body of a rule (see Definition 7).

Definition 19 (extended logic program) *An* extended logic program $\mathcal{P}$ *is a finite set of rules of the form*

$$h \leftarrow a_1, \ldots, a_n, \sim b_1, \ldots \sim b_m$$

where $h, a_1, \ldots a_n, b_1, \ldots, b_m \in \mathfrak{Lit}$. *If the body is empty, then it is called a* fact *abbreviated* h *instead of* $h \leftarrow$.

Given a set $X \subseteq \mathfrak{Lit}$ of literals, a rule r is *applicable* in X iff $a_1, \ldots, a_n \in X$ and $b_1, \ldots, b_m \notin X$. The rule r is *satisfied* by X iff it is applicable and $head(r) \in X$, or r is not applicable in X. X is a model of an extended logic program $\mathcal{P}$, iff all rules of $\mathcal{P}$ are satisfied by X. The set $X \subseteq \mathfrak{Lit}$ is *consistent* iff for every $L \in X$ it is not the case that $\neg L \in X$. An *answer set* of $\mathcal{P}$ is the least consistent set of literals that satisfies all the rules of the reduced program $\mathcal{P}$.

Definition 20 (reduct) *Let* $\mathcal{P}$ *be an extended logic program and* $X \subseteq \mathfrak{Lit}$ *a set of literals. The* GL-reduct *of* $\mathcal{P}$ *w.r.t.* X, *denoted as* $\mathcal{P}^X$, *is the union of all rules* $h \leftarrow a_1, \ldots, a_n$, *where* $(h \leftarrow a_1, \ldots, a_n, \sim b_1, \ldots, \sim b_m) \in \mathcal{P}$ *and* $X \cap \{b_1, \ldots, b_m\} = \emptyset$.

For any extended logic program $\mathcal{P}$ and a set of literals X, the GL reduct $\mathcal{P}^X$ is a logic program without default negation and therefore has a minimal model. If $\mathcal{P}^X$ is inconsistent, then its unique model is defined to be $\mathfrak{Lit}$.

Definition 21 (answer set) *Let* $\mathcal{P}$ *be an extended logic program. A consistent set of literals* $S \subseteq \mathfrak{Lit}$ *is an* answer set *of* $\mathcal{P}$, *iff* S *is the minimal model of* $\mathcal{P}^S$.

Embedding DeLP in ASP

This section introduces the main result of this paper: Embedding of defeasible logic programs into ASP extended logic programs. As a basis we use the theoretical result established in Theorem 1 above. First, we instantiate a version of Theorem 1 for DeLP with an empty preference relation and subsequently we introduce two translations from DeLP to ASP which yield admissible, resp. preferred extensions of the original *delp*.

Model theoretic characterization of DeLP

In the following we assume DeLP argumentation frameworks with an empty preference relation.

Definition 22 (plain DeLPAF) *Let $\mathcal{P} = (\Pi, \Delta)$ be a delp and $DeLPAF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq, \prec \rangle$ be its associated DeLP argumentation framework. We say that $DeLPAF$ is a* plain *DeLP argumentation framework iff $\prec = \emptyset$.*

The introduced restriction to deal only with plain DeLP argumentation frameworks has important consequences on properties of the corresponding argument sets.

Proposition 2 *Let $DeLPAF$ be a plain DeLP argumentation framework of a* delp *$\mathcal{P}$. Each acceptable argumentation line in $DeLPAF$ has a length of either 0, or 1.*

Proof. Let λ be an acceptable argumentation line for $\langle \mathcal{A}_0, a_0 \rangle$. Since the relation of preference on arguments $\succ$ is empty, the argument $\langle \mathcal{A}_0, a_0 \rangle$ can not be properly defeated. Therefore, there is either no defeater for $\langle \mathcal{A}_0, a_0 \rangle$, or there exists a blocking defeater $\langle \mathcal{A}_1, a_1 \rangle$ of $\langle \mathcal{A}_0, a_0 \rangle$ in λ. However, since λ is acceptable, according to Definition 15, there cannot be any further blocking defeater of $\langle \mathcal{A}, a_1 \rangle$ in λ. If there is no defeater for $\langle \mathcal{A}_0, a_0 \rangle$ λ contains only $\langle \mathcal{A}_0, a_0 \rangle$, i.e. has length 0. Otherwise, $\langle \mathcal{A}_0, a_0 \rangle$ is defeated by a single undefeated blocking defeater $\langle \mathcal{A}_1, a_1 \rangle$, i.e. λ has length 1.□

Corollary 4 *Each dialectical tree of a plain DeLP argumentation framework has a maximal depth 1.*

Proposition 3 *Let $\mathcal{P}$ be a* delp *and $DeLPAF$ an associated plain argumentation framework. $\langle \mathcal{A}, a \rangle$ is an argument structure for a warranted literal a in $\mathcal{P}$ if and only if there exists no defeater of $\langle \mathcal{A}, a \rangle$ in $DeLPAF$.*

Proof.

- **($\Longrightarrow$)** $\mathcal{A}$ is a warrant for a in P, if and only if there exists a dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, a \rangle}$, such that the root $\langle \mathcal{A}, a \rangle$ is marked U as *undefeated* (see Definition 18). Therefore all its child nodes in $\mathcal{T}^*_{\langle \mathcal{A}, a \rangle}$ have to be marked D as *defeated*. According to Corollary 4, all the children of the root node in a dialectical tree have to be leaves. However, according to Definition 17, the leaf nodes of a dialectical tree have to be marked U as *undefeated*. Hence the root node of a dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, a \rangle}$ of a warranted literal a cannot have any child nodes (defeaters of $\langle \mathcal{A}, a \rangle$).
- **($\Longleftarrow$)** Since $\langle \mathcal{A}, a \rangle$ has no defeaters, any dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, a \rangle}$ for $\langle \mathcal{A}, a \rangle$ has just a single node $\langle \mathcal{A}, a \rangle$. $\langle \mathcal{A}, a \rangle$ is a leaf node, hence, according to Definition 17, it is marked U as *undefeated*. But because $\langle \mathcal{A}, a \rangle$ is also the root node of $\mathcal{T}^*_{\langle \mathcal{A}, a \rangle}$ (marked *undefeated*), $\mathcal{A}$ is a warrant for a.

□

Finally, before introducing an instantiation of the Theorem 1, which provides a basis for embedding the DeLP argumentation framework into the framework of Answer Set Programming, observe a property of conflicting arguments in DeLPAF and some of its corollaries.

Proposition 4 *Let $\mathcal{P}$ be a* delp*, DeLPAF its associated plain argumentation framework and $\langle \mathcal{A}, a \rangle$, $\langle \mathcal{B}, b \rangle$ be argument structures in it. If $\langle \mathcal{A}, a \rangle$ does not attack $\langle \mathcal{B}, b \rangle$, but at the same time $\mathcal{A} \cup \mathcal{B} \cup \Pi \mathrel{|\!\!\sim} \bot$, then there exists a subargument structure $\langle \mathcal{A}', a' \rangle \sqsubseteq \langle \mathcal{A}, a \rangle$, s.t. $\langle \mathcal{A}', a' \rangle$ is a defeater of $\langle \mathcal{B}, b \rangle$.*

Proof. Provided that $\mathcal{A} \cup \mathcal{B} \cup \Pi \mathrel{|\!\!\sim} \bot$, there must be two conflicting literals a' and b', s.t. $\mathcal{A}' \subset \mathcal{A}$ is an argument for a' and $\mathcal{B}' \subseteq \mathcal{B}$ is an argument for b' and a' and b' disagree, i.e. $\{a', b'\} \cup \Pi \mathrel{|\!\!\sim} \bot$. Hence $\langle \mathcal{A}', a' \rangle \mathbf{R} \langle \mathcal{B}', b' \rangle$ and therefore also $\langle \mathcal{A}', a' \rangle \mathbf{R} \langle \mathcal{B}, b \rangle$. □

The Proposition 4 establishes a kind of weak symmetry of attacking of an argument by a preferred extensions. This observation is used to introduce the two following corollaries.

Corollary 5 *In a plain DeLPAF, each maximal conflict-free set of arguments is a preferred extension.*

Proof. Suppose S is a maximal conflict-free set of arguments which is not admissible. Then there must be an argument $\mathcal{A} \in S$, s.t. $\mathcal{A}$ is not acceptable w.r.t. S, i.e. there exists an argument $\mathcal{B} \in \mathbb{U} \setminus S$ and $\mathcal{B} \mathbf{R} \mathcal{A}$, but S does not attack $\mathcal{B}$. However, according to the Proposition 4 we have that there must be some subargument of $\mathcal{A}$, which also belongs to S (S is maximal) which attacks B, what contradicts the assumption. □

Another interesting corollary of the Proposition 4 is that in plain DeLP, the notions of preferred and stable extensions coincide.

Corollary 6 *Let $AF = \langle \mathbb{U}, \mathbf{R}, \sqsubseteq \rangle$ be a plain DeLP argumentation framework. S is a stable extension of $DeLPAF$ if and only if S is also its preferred extension.*

Proof. The proof follows a simillar argument as the proof of the Corollary 5. □

Theorem 2 *Let $\mathcal{P}$ be a* delp *and $DeLPAF$ be its associated plain argumentation framework. Let also $\langle \mathcal{A}, a \rangle$ be an argument structure in $DeLPAF$ and $\mathcal{S}^*$ be the set of preferred extensions of $DeLPAF$.*

Then a is a warranted literal in $DeLPAF$ if and only if $\mathcal{A}$ is undisputed w.r.t. $DeLPAF$.

Proof. The proof follows from Theorem 1 introduced above. First we show that the conditions imposed on $DeLPAF$ by Theorem 1 are satisfied. $DeLPAF$ attack relation $\mathbf{R}$ is irreflexive because of the Definition 13 of DeLP defeater. The proof in both directions follows in steps:

1. a is a warranted literal **iff** there exists an undefeated argument $\mathcal{A}$ supporting a. However, according to Corollary 4 and the Proposition 3, the dialectical tree $\mathcal{T}_{\langle \mathcal{A}, a \rangle}$ has to have a depth equal to 0.
2. According to Theorem 1, $\overline{\mathcal{S}}^{\mathcal{A}}$ contains only argument structures which are in conflict with $\langle \mathcal{A}, a \rangle$, i.e. if $\langle \mathcal{B}, b \rangle \in \overline{\mathcal{S}}^{\mathcal{A}}$, then either $\langle \mathcal{B}, b \rangle \mathbf{R} \langle \mathcal{A}, a \rangle$, or $\langle \mathcal{A}, a \rangle \mathbf{R} \langle \mathcal{B}, b \rangle$. In the

first case $\langle\mathcal{B}, b\rangle$ would be a defeater of $\langle\mathcal{A}, a\rangle$. In the second, according to the Proposition 4 there must be a subargument of $\langle\mathcal{B}', b'\rangle \sqsubseteq \langle\mathcal{B}, b\rangle$ which attacks $\langle\mathcal{A}, a\rangle$ and therefore $\langle\mathcal{B}', b\rangle \in \overline{\mathcal{S}}^{\mathcal{A}}$.

From that we have that a has no defeaters **iff** $\overline{\mathcal{S}}^{\mathcal{A}} = \emptyset$.

3. That means that from the set of all literals (note, that for each literal which can be derived from $\mathcal{P}$, there is an argument supporting it), we subtracted all the preferred extensions of DeLPAF which holds **iff** a belonged to all of them, i.e. it is undisputed.

□

A weaker version of Theorem 2, similar to the Corollary 2, will turn out to be useful later as well.

Theorem 3 *Let $\mathcal{P}$ be a* delp *and $DeLPAF$ be its associated plain argumentation framework. Let also $\langle\mathcal{A}, a\rangle$ be an argument structure in $DeLPAF$ and $\mathcal{S}$ be the set of all admissible sets in it, containing also all the preferred extensions of $DeLPAF$.*

Then a is a warranted literal w.r.t. $DeLPAF$ if and only if $\overline{\mathcal{S}}^{\mathcal{A}} = \emptyset$.

Proof. The proof follows from Corollary 3 and the proof of Theorem 2, step 2, above. □

Translating DeLP in ASP

Theorems 2 and 3 establish a generic scheme for embedding the DeLP framework without preferences into ASP. Provided a *delp* $\mathcal{P}$ with a corresponding plain argumentation framework, the scheme follows a three step algorithm:

1. Translate a given *delp* $\mathcal{P}$ in a corresponding extended logic program,
2. by computing its answers sets, obtain an appropriate set of literals corresponding to a well-defined notion in the $DeLPAF$ theory introduced above, and finally
3. extract the set of warranted literals from the obtained answer sets.

Below, we introduce two translations of defeasible logic programs into extended logic programs answer sets of which enjoy favourable properties w.r.t. the argumentation framework theory. First, however we introduce a supplementary notion of an argument set completeness.

Definition 23 (argument set completion) *We say that a set of argument structures S is* complete *w.r.t. an argumentation framework $AF = \langle\mathbb{U}, \mathbf{R}, \sqsubseteq\rangle$ iff for every $\langle\mathcal{A}, h\rangle \in \mathbb{U}$ holds: If $\mathcal{A} \subseteq \bigcup_{\mathcal{A}' \in S} \mathcal{A}'$ then $\langle\mathcal{A}, h\rangle \in S$.*

Note that every preferred extension of an argumentation framework is complete.

Definition 24 (vanilla translation) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp *The resulting extended logic program $\mathfrak{Trans}(\mathcal{P})$ is constructed as follows:*

1. For each $r \in \Delta$, $\mathfrak{Trans}(\mathcal{P})$ contains the following rules

$$block(l_r) \leftarrow \sim \neg block(l_r) \tag{1}$$

$$\neg block(l_r) \leftarrow \sim block(l_r) \tag{2}$$

$$head(r) \leftarrow body(r), \sim block(l_r) \tag{3}$$

where l_r is a new unique literal corresponding to the rule r.

2. Π is copied to $\mathfrak{Trans}(\mathcal{P})$, i.e. $\mathfrak{Trans}(\mathcal{P})$ contains for each $r \in \Pi$ the following rule

$$head(r) \leftarrow body(r) \tag{4}$$

The straightforward vanilla translation $\mathfrak{Trans}$ stems from an observation, that each non-contradictory set of defeasible rules $\Delta' \subseteq \Delta$, s.t. $\Pi \cup \Delta' \not\mathrel{|\!\!\sim} \bot$, of a *delp* $\mathcal{P} = (\Pi, \Delta)$ models some complete admissible set of arguments (all arguments which can be constructed from Δ'), as well as for each complete admissible set of arguments we can identify a set of defeasible rules $\Delta' \subseteq \Delta$ inducing it. Note however, that it might happen that some rules will always remain unused as building blocks for arguments in Δ': Consider $\Pi = \{a. b \leftarrow a.\}$ and $\Delta' = \Delta = \{b \prec a.\}$. Although $\langle\{b \prec a.\}, b\rangle$ is not minimal, the rule $b \prec a.$ is satisfied by the quasi argument structure.

The tuple of Rules 1 and 2 of Definition 24 encode a non-deterministic choice of a set of rules Δ' from Δ. Provided l_r is not blocked, i.e. $\neg block(l_r)$ holds, the Rule 3 enables firing the corresponding rule r from Δ. Finally, the Rule 4 copies strict rules from Π to the translated logic program, thus allowing an equivalent derivation as in the original *delp*.

The answer set semantics then computes a set of literals corresponding to completions of admissible sets of arguments built from such choices of Δ'. The following proposition articulates the relation between answer sets of the resulting logic program and the plain argumentation framework corresponding to the *delp* $\mathcal{P}$.

Proposition 5 *Let $\mathcal{P}$ be a* delp *and let $DeLPAF$ be its associated plain argumentation framework.*

Then there exists an answer set M of $\mathfrak{Trans}(\mathcal{P})$ if and only if there exists a complete admissible set S of $DeLPAF$, s.t. $\mathfrak{Lit}(S) = M \cap \mathfrak{Lit}(\mathcal{P})$.

Proof.

($\Longrightarrow$) Let M be an answer set of $\mathfrak{Trans}(\mathcal{P})$ and $\Delta' = \{r \in \Delta \mid M \not\models block(l_r)\}$. According to Definition 21, M is the least model of the reduct $\mathfrak{Trans}(\mathcal{P})^M = \Pi \cup \Delta' \cup \{block(L) \mid M \not\models \neg block(L)\} \cup \{\neg block(L) \mid M \not\models block(L)\}$. Therefore $M \cap \mathfrak{Lit}(\mathcal{P})$ is the least model of $\Pi \cup \Delta'$.

Let $S = \{\langle\mathcal{A}, h\rangle \mid \mathcal{A} \subseteq \Delta'\}$. If $\langle\mathcal{A}, h\rangle$ is an argument structure, such that $\mathcal{A} \subseteq \bigcup_{\mathcal{A}' \in S} \mathcal{A}'$, then $\mathcal{A} \subseteq \Delta'$ and $\langle\mathcal{A}, h\rangle \in S$. Thus S is complete. Because $M \cap \mathfrak{Lit}(\mathcal{P})$ is a model of $\Pi \cup \Delta'$, S is also conflict-free. Each conflict-free set of argument structures of a plain argumentation framework is an admissible set. Hence S is a complete amissible set of $DeLPAF$.

Let $\mathcal{A} \subseteq \Delta'$ be an argument for h. Then there exists a derivation $L_1, \ldots, L_n$, $n \geq 1$ of h from $\Pi \cup \Delta'$. By mathematical induction on n we can prove that every model of $\Pi \cup \Delta'$ satisfies also $L_1, \ldots, L_n = h$. Therefore $\mathfrak{Lit}(S) \subseteq M \cap \mathfrak{Lit}(P)$. Let h be a literal in the least model of $\Pi \cup \Delta'$. Then there exists a derivation $L_1, \ldots, L_n$, $n \geq 1$ of h from $\Pi \cup \Delta'$. If we take the derivation with minimal set $\mathcal{A}$ of used defeasible rules, we get an argument $\mathcal{A}$ for h. Therefore $\mathfrak{Lit}(S) \supseteq M \cap \mathfrak{Lit}(P)$.

($\Longleftarrow$) Let S be a complete admissible set of argument structures of $DeLPAF$ and $\Delta' = \{r \in \Delta \mid \mathfrak{Lit}(S) \models r\}$. Because S is closed, $\mathfrak{Lit}(S)$ is the least model of $\Pi \cup \Delta'$. Let $M = \mathfrak{Lit}(S) \cup \{block(l_r) \mid r \notin \Delta'\} \cup \{\neg block(l_r) \mid r \in \Delta'\}$. We show that M is the least model of the reduct $\mathfrak{Trans}(\mathcal{P})^M = \Pi \cup \Delta' \cup \{block(l_r) \mid r \notin \Delta'\} \cup \{\neg block(l_r) \mid r \in \Delta'\}$, i.e. M is an answer set of $\mathfrak{Trans}(\mathcal{P})$.

It is easy to see that M is the least model of $\mathfrak{Trans}(\mathcal{P})^M$ if and only if $\mathfrak{Lit}(S)$ is the least model of $\Pi \cup \Delta'$. Let $h \in \mathfrak{Lit}(S)$. Then there exists a derivation of h from $\Pi \cup \Delta'$. Therefore every model of $\Pi \cup \Delta'$ must also satisfy h, i.e. the least model of $\Pi \cup \Delta'$ contains $\mathfrak{Lit}(S)$. $\mathfrak{Lit}(S)$ satisfies Δ'. Because it is complete, it also satisfies Π. Thus $\mathfrak{Lit}(S)$ is the least model of $\Pi \cup \Delta'$.

□

The second translation of defeasible logic programs to extended logic programs aims at significantly reducing the number of yielded answer sets so, that they uniquely correspond to the set of preferred extensions of the original *delp*. It further builds on the vanilla translation and adds additional filtering for all the non-preferred admissible sets.

Definition 25 (stable translation) *Let $\mathcal{P} = (\Pi, \Delta)$ be a* delp. *The resulting extended logic program $\mathfrak{Trans}^*(\mathcal{P})$ contains all the rules from $\mathfrak{Trans}(\mathcal{P})$ and*

1. *for each $r_1 \in \Pi$ and $r_2 \in \Delta$, $\mathfrak{Trans}^*(\mathcal{P})$ contains the following rules*
$$check(head(r_1), head(r_2)) \leftarrow block(l_{r_2}), check(body(r_1), head(r_2)) \tag{1}$$
$$fail(head(r_2)) \leftarrow block(l_{r_2}), check(head(r_1), head(r_2)), check(\neg head(r_1), head(r_2)) \tag{2}$$
2. *for each $r_1 \in \Delta$ and $r_2 \in \Delta$, $\mathfrak{Trans}^*(\mathcal{P})$ contains the following rules*
$$check(head(r_1), head(r_2)) \leftarrow block(l_{r_2}), check(body(r_1), head(r_2)), \sim block(l_{r_1}) \tag{3}$$
$$fail(head(r_2)) \leftarrow block(l_{r_2}), check(head(r_1), head(r_2)), check(\neg head(r_1), head(r_2)) \tag{4}$$
3. *for each $r \in \Delta$, $\mathfrak{Trans}^*(\mathcal{P})$ contains the following rules*
$$check(head(r), head(r)) \leftarrow block(l_r) \tag{5}$$
$$\leftarrow block(l_r), \sim fail(head(r)) \tag{6}$$

where $check(\{L_1, \ldots, L_n\}, L)$, $n \geq 0$, denotes the set of literals $check(L_1, L), \ldots, check(L_n, L)$.

The filtering of non-preferred admissible sets in the translation $\mathfrak{Trans}^*$ above tests whether by adding the head of a blocked rule, a conflict really arises, i.e. we have a maximal consistent set of literals.

Extended logic programs produced by the stable translation $\mathfrak{Trans}^*$ thus yield answer sets, which after reduction to the language of the original *delp* $\mathcal{P}$ uniquely correspond to preferred extensions of $\mathcal{P}$. This relationship is articulated formally by the following proposition.

Proposition 6 *Let $\mathcal{P}$ be a* delp *and let $DeLPAF$ be its associated plain argumentation framework.*

Then there exists an answer set M of $\mathfrak{Trans}^*(\mathcal{P})$ if and only if there exists a preferred extension S of $DeLPAF$, s.t. $\mathfrak{Lit}(S) = M \cap \mathfrak{Lit}(\mathcal{P})$.

Proof.

($\Longrightarrow$) Let M be an answer set of $\mathfrak{Trans}^*(\mathcal{P})$ and $\Delta' = \{r \in \Delta \mid M \not\models block(l_r)\}$. Let $S = \{\langle \mathcal{A}, h\rangle \mid \mathcal{A} \subseteq \Delta'\}$. Similarly as in the proof of the proposition 5, S is conflict-free. Let $\langle \mathcal{A}, h\rangle \notin S$. Then there exists a rule $r \in \mathcal{A} \setminus \Delta'$ such that $M \not\models r$. M satisfies $block(l_r)$ because of the rule (3) in Definition 24 and M satisfies $fail(head(r))$ because of the rule (3) in Definition 25. In addition $M \models check(L, head(r))$ if and only if there exists a derivation of L from $\Pi \cup \Delta' \cup \{head(r)\}$ (see the rules (1), (2) and (3) from the definition 25). Because $M \models fail(head(r))$, $\Pi \cup \Delta' \cup \{head(r)\}$ is contradictory (see the rules (1) and (2) from the definition 25). Thus S is a maximal conflict-free set of argument structures, i.e. according to the Corollary 5 a preferred extention of DeLPAF.

($\Longleftarrow$) Let S be a preferred extension of $\mathcal{P}$ and $\Delta' = \{r \in \Delta \mid \mathfrak{Lit}(S) \models r\}$. According to the proof of Proposition 5, $M' = \mathfrak{Lit}(S) \cup \{block(l_r) \mid r \notin \Delta'\} \cup \{\neg\, block(l_r) \mid r \in \Delta'\}$ is an answer set of $\mathfrak{Trans}(\mathcal{P})$. Because S is a maximal conflict-free set of argument structures, then for every rule $r \notin \Delta'$ holds $\Pi \cup \Delta' \cup \{head(r)\} \mathrel{|\!\!\sim} \bot$. Therefore $M = M' \cup \{fail(head(r)) \mid r \notin \Delta'\} \cup \bigcup_{r \notin \Delta'} \{check(L, head(r)) \mid \Pi \cup \Delta' \cup \{head(r)\} \mathrel{|\!\!\sim} L\}$ is the least model of $\mathfrak{Trans}_*(\mathcal{P})^M$, i.e. it is an answer set of $\mathfrak{Trans}^*(\mathcal{P})$.

□

Example 1 *Let $\mathcal{P} = (\Pi, \Delta)$ be the following* delp*:*

$$\Pi: \quad \begin{array}{rcl} a & & \\ g & \leftarrow & b \\ h & \leftarrow & c \\ \neg h & \leftarrow & c \end{array} \qquad \Delta: \quad \begin{array}{rcl} b & \prec & a \\ c & \prec & a \end{array}$$

Listings 1 and 2 show the extended logic programs resulting from the vanilla and the stable translation.

The vanilla translation $\mathfrak{Trans}(\mathcal{P})$ has two answer sets $M_1 = \{a, b, g, \neg block(l_{r_1}), block(l_{r_2})\}$ and $M_2 = \{a, block(l_{r_1}), block(l_{r_2})\}$. The interpretation $M_3 = M_1 \cup \{check(c, c), check(h, c), check(\neg h, c), fail(c)\}$ is the only answer set of the stable translation $\mathfrak{Trans}^*(\mathcal{P})$.

Warranted literals extraction

Using the previously introduced theoretical results, we can finally conclude the embedding of DeLP to ASP and propose algorithms for extraction of warranted literals from answer sets of a translated defeasible logic program. The straightforward scheme follows three main steps: (1) DeLP to ASP

Listing 1 Example of a vanilla translation.

```
% Translation rules (1) and (2)
 block(l_r1) ← ∼ ¬block(l_r1)        block(l_r2) ← ∼ ¬block(l_r2)
¬block(l_r1) ← ∼block(l_r1)         ¬block(l_r2) ← ∼block(l_r2)

% Translation rule (3)               % Copy of Π. Translation rule (4)
b ← a, ∼block(l_r1)                  a.          h ← c
c ← a, ∼block(l_r2)                  g ← b.     ¬h ← c
```

Listing 2 Example of a stable translation.

```
% Translation rule (1)
check(a, b) ← block(l_r1)
check(a, c) ← block(l_r2)
check(g, b) ← block(l_r1), check(b, b)
check(g, c) ← block(l_r2), check(b, c)
check(h, b) ← block(l_r1), check(c, b)
check(h, c) ← block(l_r2), check(c, c)
check(¬h, b) ← block(l_r1), check(c, b)
check(¬h, c) ← block(l_r2), check(c, c)

% Translation rule (3)
check(b, c) ← block(l_r2), check(a, c), ∼block(l_r1)
check(c, b) ← block(l_r1), check(a, b), ∼block(l_r2)

% Translation rules (2) and (4)
fail(b) ← block(l_r1), check(a, b), check(¬a, b)
fail(c) ← block(l_r2), check(a, c), check(¬a, c)
fail(b) ← block(l_r1), check(b, b), check(¬b, b)
fail(c) ← block(l_r2), check(b, c), check(¬b, c)
fail(b) ← block(l_r1), check(c, b), check(¬c, b)
fail(c) ← block(l_r2), check(c, c), check(¬c, c)
fail(b) ← block(l_r1), check(g, b), check(¬g, b)
fail(c) ← block(l_r2), check(g, c), check(¬g, c)
fail(b) ← block(l_r1), check(h, b), check(¬h, b)
fail(c) ← block(l_r2), check(h, c), check(¬h, c)

% Translation rule (5)
check(b, b) ← block(l_r1)
check(c, c) ← block(l_r2)

% Translation rule (6)
← block(l_r1), ∼fail(b)
← block(l_r2), ∼fail(c)
```

translation, (2) answer set computation, and, finally, (3) warranted literals extraction.

Depending on which translation from DeLP to ASP is used, different algorithm for extraction of warranted literals must be employed. In the case of the stable translation $\mathfrak{Trans}^*$ (Definition 25), according to Proposition 6, the answer sets of a program $\mathfrak{Trans}^*(\mathcal{P})$ correspond to the set of preferred extensions of the *delp* $\mathcal{P}$. To extract the set of warranted literals from the answer sets, we can employ Theorem 2, i.e. the set of literals corresponds to the intersection of all the answer sets. Algorithm 1 displays the pseudocode of the described warranted literals extraction.

For the vanilla translation $\mathfrak{Trans}$, according to Proposition 5, the set of answer sets of a program $\mathfrak{Trans}(\mathcal{P})$ correspond to the set of all complete admissible sets of the *delp* $\mathcal{P}$. Then for each literals h, we can check whether it is warranted w.r.t. the resulting answer sets by employing Theorem 3, i.e. from the set of all literals, we subtract those answer sets to which a literal h in consideration belongs. If the resulting set of literals is empty, then according to Theorem 3, h is warranted. The Algorithm 2 displays a pseudocode for extraction of warranted literals exploiting the vanilla transformation $\mathfrak{Trans}$ and Theorem 3.

Note, that Algorithm 2 based on Corollary 2, the weaker version of Theorem 1, is very generic. It would work also for any translation from DeLP to ASP producing an extended logic program yielding answer sets corresponding to a mixture of preferred extensions and other admissible sets. Indeed, it would also work if we would replace the vanilla transformation $\mathfrak{Trans}$ in it by the stable transformation $\mathfrak{Trans}^*$.

As far as the time complexity of the introduced algorithms is concerned, the most time consuming component of the three stage algorithm is computation of answer sets of the translated extended logic program. While the warranted literals extraction yields a rather straightforward algorithm of polynomial complexity, the translation is either linear, in the case of the vanilla translation, or quadratic for the stable translation. Note however, that because of the linear space complexity of the resulting extended logic program, the vanilla translation is *modular*: By adding a new rule r to the original *delp* $\mathcal{P}$, it suffices to translate only the new rule to obtain the resulting extended logic program, i.e. $\mathfrak{Trans}(\Pi \cup \{r\}) = \mathfrak{Trans}(\Pi) \cup \mathfrak{Trans}(\{r\})$. The same property does not hold for the stable translation, which on the other hand yields logic programs with significantly fewer answer sets w.r.t. those produced by the vanilla translations.

Discussion & Conclusion

In this paper we discussed embeddings of the argumentation framework of *Defeasible Logic Programming* into *Answer Set Programming*. We first provided several theoretical results about Dung's abstract argumentation framework culminating in the introduction of Theorem 1 and its weaker version, Corollary 2. Then, we instantiated these results in the concrete argumentation approach of DeLP and finally we used them to show how warranted literals of a given *delp* program can be computed by (1) *translating the program* to extended logic program, subsequently (2) *processing it by the answer set semantics* and (3) we introduced *warranted literals extraction algorithms* for the corresponding DeLP-2-ASP translations. The main consequence of the proposed scheme is a practical result:

> *Our results enable to use state-of-the-art ASP solvers for answering queries of DeLP.*

We introduced two extreme translations from DeLP to ASP: The straightforward vanilla translation $\mathfrak{Trans}$ and the stable $\mathfrak{Trans}$ yielding a large and a smallest number of answer sets respectively. But our approach enables a whole class of translations between the two frameworks. The only requirement is, that the answer sets of the resulting extended logic

Algorithm 1 Algorithm for extracting warranted literals of a *delp* exploiting translation to ASP $\mathfrak{Trans}^*$.

procedure ExtractWarrants*(*delp* $\mathcal{P}$)
  compile $\mathcal{P}$ to $\mathfrak{Trans}^*(\mathcal{P})$
  $\mathcal{S}^*$:= compute the answer sets of $\mathfrak{Trans}^*(\mathcal{P})$
  $W := \bigcap_{S \in \mathcal{S}^*} S$
end procedure $\triangleright$ W contains all warranted literals of $\mathcal{P}$

Algorithm 2 Algorithm for extracting warranted literals using $\mathfrak{Trans}$.

procedure ExtractWarrants(*delp* $\mathcal{P}$)
  compile $\mathcal{P}$ to $\mathfrak{Trans}(\mathcal{P})$
  $\mathcal{S}$:= compute the answer sets of $\mathfrak{Trans}(\mathcal{P})$
  $W := \emptyset$
  for each $h \in \mathfrak{Lit}$ **do**
    if $[\mathbb{U} \setminus (\bigcup_{\substack{S \in \mathcal{S} \\ h \in S}} S)] = \emptyset$ **then**
      $W := W \cup h$
    end if
  end for
end procedure $\triangleright$ W contains all warranted literals of $\mathcal{P}$

program contain the preferred extensions of the original program among other admissible sets.

As (Thimm & Kern-Isberner 2008) points out, there is not much work reported on links between DeLP and ASP. To our knowledge, (Thimm & Kern-Isberner 2008) is the only publication on this topic to date.

Our approach provides a stronger result than that of (Thimm & Kern-Isberner 2008). It provides a basis for a *purely syntactical* compilation of defeasible logic programs into extended logic programs. The approach of (Thimm & Kern-Isberner 2008) required computation of minimal disagreement sets of the input *delp* which were subsequently used as the basis for its translation to a corresponding extended logic program.

Our result is based on the observation that from a set of preferred extensions of an argumentation framework with an appropriate attack relation, we can extract the set of defeaters, to a given argument $\mathcal{A}$. Subsequently, we use this observation to show, that in the case of an empty preference relation $\prec$ on arguments, an argument is a warrant if and only if the set of its defeaters is empty. The introduced translations of *delp* to extended logic programs are based on the idea, that answer sets of the resulting logic program correspond to preferred extensions, resp. admissible sets of the original *delp* among which all the preferred extensions are present. From the set of answer sets, the set of warranted literals can be extracted.

Moreover, note that the general Theorem 1 can be used also in the context of DeLP framework with a non-empty preference relation for constructing a dialectical tree of a query: Acceptable argumentation lines are constructed by interleaving arguments in favour and against the query. By exploiting the set theoretic relations between admissible sets it should be fairly simple to search for defeaters of an argument in question.

In this paper we established a strong link between DeLP and ASP frameworks in the sense that we provided a set of sufficient and necessary requirements on answer sets of extended logic programs resulting from a purely syntactical translation from defeasible logic programs. Our result can be used to obtain implemented defeasible argumentation systems which instead of DeLP solvers, exploit the full power of state-of-the-art ASP solvers.

In our future work, we will try to experimentally evaluate our approach to computing warranted literals of defeasible logic programs and compare them with the existing DeLP solvers. Additionally, we will work towards lifting our results to DeLP frameworks with non-empty preference relation. On a parallel track we are planning (using the results of this paper) to relate the theoretical results regarding updates (e.g. (Rotstein *et al.* 2008)) and evolutions of argumentation systems, to those from the field of logic program updates, such as (Leite 2003).

References

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Besnard, P., and Hunter, A. 2000. Towards a logic-based theory of argumentation. In *AAAI/IAAI*, 411–416. AAAI Press / The MIT Press.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–358.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *TPLP* 4(1-2):95–138.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *ICLP/SLP*, 1070–1080.

Leite, J. A. 2003. *Evolving Knowledge Bases*, volume 81 of *Frontiers of Artificial Intelligence and Applications*. IOS Press.

Prakken, H., and Vreeswijk, G. 2002. Logics for defeasible argumentation. In *Handbook on Philosophical Logic*, volume 4, 2nd edition. Kluwer Academic Publishers, Dordrecht. 218–319.

Rotstein, N. D.; Moguillansky, M. O.; Falappa, M. A.; García, A. J.; and Simari, G. R. 2008. Argument Theory Change: Revision Upon Warrant. In *Proceedings of International Conference on Computational Models of Argument, COMMA 2008*.

Thimm, M., and Kern-Isberner, G. 2008. On the relationship of defeasible argumentation and answer set programming. In *Proceedings of the Second International Conference on Computational Models of Argument, COMMA'08*, 393–404.