Pruning Search Space in Defeasible Argumentation

Carlos Iván Chesñevar Guillermo Ricardo Simari Alejandro Javier García

Universidad Nacional del Sur, Av. Alem 1253, (8000) Baha Blanca, ARGENTINA
Email: {cic,grs,ajg}@cs.uns.edu.ar

Abstract. Defeasible argumentation has experienced a considerable growth in AI in the last decade. Theoretical results have been combined with development of practical applications in AI & Law, Case-Based Reasoning and various knowledge-based systems. However, the dialectical process associated with inference is computationally expensive. This paper focuses on speeding up this inference process by pruning the involved search space. Our approach is twofold. On one hand, we identify distinguished literals for computing defeat. On the other hand, we restrict ourselves to a subset of all possible conflicting arguments by introducing dialectical constraints. This approach can be adapted to many existing argumentation systems.

1 Preliminaries

Argumentation systems (AS) have emerged during the last decade as a promising formalization of defeasible reasoning [SL92,PV99,KT99]. Starting from the non-monotonic reasoning community, AS evolved and matured within several areas of Computer Science such as AI & Law, knowledge representation, default reasoning and logic programming.

The inference process in AS is computationally expensive when compared with alternative frameworks for modeling commonsense reasoning, such as traditional rule-based systems. This paper discusses theoretical considerations that lead to obtain efficient implementations of AS. As a basis for our analysis we use *defeasible logic programming* [Gar97,GSC98]. The paper is structured as follows: section 2 introduces *defeasible logic programming*. Section 3 presents the main contributions of the paper. Finally, section 4 concludes.

2 Defeasible Logic Programming: fundamentals

Defeasible Logic Programming (*DeLP*) is a logic programming formalism which relies upon defeasible argumentation for solving queries. The *DeLP* language [SL92,Gar97,GSC98] is defined in terms of two disjoint sets of rules: *strict rules* for representing strict (sound) knowledge, and *defeasible rules* for representing tentative information. Rules will be defined using *literals*. A literal L is an atom p or a negated atom $\sim p$, where the symbol "$\sim$" represents *strong negation*.

Definition 2.1 (Strict and Defeasible Rules). *A* strict rule *(*defeasible rule*) is an ordered pair, conveniently denoted by* $Head \leftarrow Body$ *(*$Head \prec Body$*), whose first member, Head, is a literal, and whose second member, Body, is a finite set of literals. A strict rule (defeasible rule) with the head* L_0 *and body* $\{L_1, \ldots, L_n\}$ *can also be written as* $L_0 \leftarrow L_1, \ldots, L_n$ *(*$L_0 \prec L_1, \ldots, L_n$*). If the body is empty, it is written* $L \leftarrow true$ *(*$L \prec true$*), and it is called a* fact *(*presumption*). Facts may also be written as* L*.*

Definition 2.2 (Defeasible Logic Program $\mathcal{P}$). *A* defeasible logic program *(dlp) is a finite set of strict and defeasible rules. If* $\mathcal{P}$ *is a dlp, we will distinguish in* $\mathcal{P}$ *the subset* Π *of strict rules, and the subset* Δ *of defeasible rules. When required, we will denote* $\mathcal{P}$ *as* (Π, Δ)*.*

Example 2.3. Consider an intelligent agent which has to control an engine whose performance is determined by three switches $sw1$, $sw2$ and $sw3$.[1] The switches regulate different features of the engine's behavior, such as pumping system and working speed. We can model the engine behavior using a *dlp* program (Π, Δ), where $\Pi = \{(sw1 \leftarrow), (sw2 \leftarrow), (sw3 \leftarrow), (heat \leftarrow), (\sim fuel\_ok \leftarrow pump\_clogged)\}$ (specifying that the three switches are on, there is heat, and whenever the pump gets clogged, fuel is not ok), and Δ models the possible behavior of the engine under different conditions (see figure 1).

Given a *dlp* $\mathcal{P}$, a *defeasible derivation* for a query q is a finite set of rules obtained by backward chaining from q (as in a PROLOG program) using both strict and defeasible rules from $\mathcal{P}$. The symbol "$\sim$" is considered as part of the predicate when generating a defeasible derivation. A set of rules $\mathcal{S}$ is *contradictory* iff there is a defeasible derivation from $\mathcal{S}$ for some literal p and its complement $\sim p$. Given a *dlp* $\mathcal{P}$, we will assume that its set Π of strict rules is non-contradictory.[2]

Definition 2.4 (Defeasible Derivation Tree). *Let* $\mathcal{P}$ *be a dlp, and let* h *be a ground literal. A* defeasible derivation tree T *for* h *is a finite tree, where all nodes are labelled with literals, satisfying the following conditions:*

[1] For the sake of simplicity, we restrict ourselves to propositional language for this example.

[2] If a contradictory set of strict rules is used in a *dlp* the same problems as in extended logic programming would appear. The corresponding analysis has been done elsewhere [GL90].

$pump\_fuel\_ok \prec sw1$	(when sw1 is on, normally fuel is pumped properly);
$fuel\_ok \prec pump\_fuel\_ok$	(when fuel is pumped, normally fuel works ok);
$pump\_oil\_ok \prec sw2$	(when sw2 is on, normally oil is pumped);
$oil\_ok \prec pump\_oil\_ok$	(when oil is pumped, normally oil works ok);
$engine\_ok \prec fuel\_ok, oil\_ok$	(when there is fuel and oil, normally engine works ok);
$\sim engine\_ok \prec fuel\_ok, oil\_ok, heat$	(when there is fuel, oil and heat, usually engine is not working ok);
$\sim oil\_ok \prec heat$	(when there is heat, normally oil is not ok);
$pump\_clogged \prec pump\_fuel\_ok, low\_speed$	(when fuel is pumped and speed is low, there are reasons to believe that the pump is clogged);
$low\_speed \prec sw2$	(when sw2 is on, normally speed is low);
$\sim low\_speed \prec sw2, sw3$	(when both sw2 and sw3 are on, speed tends not to be low).
$fuel\_ok \prec sw3$	(when sw3 is on, normally fuel is ok).

Fig. 1. Set Δ (example 2.3)

1. *The root node of T is labelled with h.*
2. *For each node N in T labelled with the literal L, there exists a ground instance of a strict or defeasible rule $r \in \mathcal{P}$ with head L_0 and body $\{L_1, L_2, \ldots, L_k\}$ in $\mathcal{P}$, such that $L = L\sigma$ for some ground variable substitution σ, and the node N has exactly k children nodes labelled as $L_1\sigma, L_2\sigma, \ldots, L_k\sigma$.*

The sequence S=$[r_1, r_2, \ldots r_k]$ of grounded instances of strict and defeasible rules used in building T will be called a defeasible derivation of h.

Definition 2.5 (Argument/Subargument). *Given a dlp $\mathcal{P}$, an* argument *$\mathcal{A}$ for a query q, denoted $\langle \mathcal{A}, q\rangle$, is a subset of ground instances of the defeasible rules of $\mathcal{P}$, such that:*

1. *there exists a defeasible derivation for q from $\Pi \cup \mathcal{A}$ (also written $\Pi \cup \mathcal{A} \vdash q$),*
2. *$\Pi \cup \mathcal{A}$ is non-contradictory, and*
3. *$\mathcal{A}$ is minimal with respect to set inclusion.*

An argument $\langle \mathcal{A}_1, q_1\rangle$ is a sub-argument *of another argument $\langle \mathcal{A}_2, q_2\rangle$, if $\mathcal{A}_1 \subseteq \mathcal{A}_2$.*

Definition 2.6 (Counterargument / Attack). *An argument $\langle \mathcal{A}_1, q_1\rangle$* counterargues *(or* attacks*) an argument $\langle \mathcal{A}_2, q_2\rangle$ at a literal q iff there is an subargument $\langle \mathcal{A}, q\rangle$ of $\langle \mathcal{A}_2, q_2\rangle$ such that the set $\Pi \cup \{q_1, q\}$ is contradictory.*

Informally, a query q will succeed if the supporting argument is not defeated; that argument becomes a *justification*. In order to establish if $\mathcal{A}$ is a non-defeated argument, *defeaters* for $\mathcal{A}$ are considered, i. e. counterarguments that are preferred to $\mathcal{A}$ according to some preference criterion. *DeLP* considers a particular criterion called *specificity* [SL92,GSC98] which favors an argument with greater information content and/or less use of defeasible rules.[3]

Definition 2.7 (Proper Defeater / Blocking Defeater). *An argument $\langle \mathcal{A}_1, q_1\rangle$ defeats $\langle \mathcal{A}_2, q_2\rangle$ at a literal q iff there exists a subargument $\langle \mathcal{A}, q\rangle$ of $\langle \mathcal{A}_2, q_2\rangle$ such that $\langle \mathcal{A}_1, q_1\rangle$ counterargues $\langle \mathcal{A}_2, q_2\rangle$ at q, and either: (a) $\langle \mathcal{A}_1, q_1\rangle$ is "better" that $\langle \mathcal{A}, q\rangle$ (then $\langle \mathcal{A}_1, q_1\rangle$ is a proper defeater of $\langle \mathcal{A}, q\rangle$); or (b) $\langle \mathcal{A}_1, q_1\rangle$ is unrelated by the preference order to $\langle \mathcal{A}, q\rangle$ (then $\langle \mathcal{A}_1, q_1\rangle$ is a blocking defeater of $\langle \mathcal{A}, q\rangle$).*

Since defeaters are arguments, there may exist defeaters for the defeaters and so on. That prompts for a complete dialectical analysis to determine which arguments are ultimately defeated. Ultimately undefeated arguments will be labelled as *U-nodes*, and the defeated ones as *D-nodes*. Next we state the formal definitions required for this process:

Definition 2.8 (Dialectical Tree. Argumentation line). *Let $\mathcal{A}$ be an argument for q. A dialectical tree for $\langle \mathcal{A}, q\rangle$, denoted $\mathcal{T}_{\langle \mathcal{A},q\rangle}$, is recursively defined as follows:*

1. *A single node labeled with an argument $\langle \mathcal{A}, q\rangle$ with no defeaters is by itself the dialectical tree for $\langle \mathcal{A}, q\rangle$.*
2. *Let $\langle \mathcal{A}_1, q_1\rangle, \langle \mathcal{A}_2, q_2\rangle, \ldots, \langle \mathcal{A}_n, q_n\rangle$ be all the defeaters (proper or blocking) for $\langle \mathcal{A}, q\rangle$. We construct the dialectical tree for $\langle \mathcal{A}, q\rangle$, $\mathcal{T}_{\langle \mathcal{A},q\rangle}$, by labeling the root node with $\langle \mathcal{A}, q\rangle$ and by making this node the parent node of the roots of the dialectical trees for $\langle \mathcal{A}_1, q_1\rangle, \langle \mathcal{A}_2, q_2\rangle, \ldots, \langle \mathcal{A}_n, q_n\rangle$.*

A path $\lambda = [\, \langle \mathcal{A}_0, q_0\rangle, \ldots \langle \mathcal{A}_m, q_m\rangle \,]$ in $\mathcal{T}_{\langle \mathcal{A},q\rangle}$ is called argumentation line. *We will denote as $S_\lambda = \bigcup_{i=2k}\langle \mathcal{A}_i, q_i\rangle$ ($I_\lambda = \bigcup_{i=2k+1}\langle \mathcal{A}_i, q_i\rangle$) the set of all even-level (odd-level) arguments in λ. Even-level (odd-level) arguments are also called* supporting arguments *or S-arguments (*interferring *arguments or I-arguments).*

Definition 2.9 (Labelling of the Dialectical Tree). *Let $\langle \mathcal{A}, q\rangle$ be an argument and $\mathcal{T}_{\langle \mathcal{A},q\rangle}$ its dialectical tree, then:*

[3] See [GSC98] for details.

1. *All the leaves in $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ are labelled as U-nodes.*
2. *Let $\langle \mathcal{B}, h \rangle$ be an inner node of $\mathcal{T}_{\langle \mathcal{A},q \rangle}$. Then $\langle \mathcal{B}, h \rangle$ will be a U-node iff every child of $\langle \mathcal{B}, h \rangle$ is a D-node. The node $\langle \mathcal{B}, h \rangle$ will be a D-node iff it has at least one child marked as U-node.*

To avoid the occurrence of *fallacious argumentation* [SCG94], two basic additional constraints on dialectical trees are imposed on any argumentation line λ: a) there can be no repeated arguments (circular argumentation) and b) the set of all odd-level (even-level) arguments in λ should be *non-contradictory* wrt Π in order to avoid *contradictory* argumentation. Defeaters satisfying these constraints are called *acceptable.*[4]

An argument $\mathcal{A}$ which turns to be ultimately undefeated is called a *justification.*

Definition 2.10 (Justification). *Let $\mathcal{A}$ be an argument for a literal q, and let $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ be its associated acceptable dialectical tree. The argument $\mathcal{A}$ for q will be a* justification *iff the root of $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ is a U-node.*

Example 2.11. Consider example 2.3, and assume our agent is trying to determine whether the engine works ok by finding a justification supporting *engine_ok*. The set of defeasible rules $\mathcal{A} = \{$ *pump_fuel_ok* $-\!\prec$ *sw1*, *pump_oil_ok* $-\!\prec$ *sw2*, $fuel\_ok$ $-\!\prec$ *pump_fuel_ok*, *oil_ok* $-\!\prec$ *pump_oil_ok*, *engine_ok* $-\!\prec$ $fuel\_ok, oil\_ok$ $\}$. is an *argument* for *engine_ok*, i. e. , $\langle \mathcal{A}, engine\_ok \rangle$. But there exists a *counterargument* $\mathcal{B} = \{$ *pump_fuel_ok* $-\!\prec$ *sw1*, *low_speed* $-\!\prec$ *sw2*, *pump_clogged* $-\!\prec$ *pump_fuel_ok*, *low_speed* $\}$ which supports the conclusion $\sim fuel\_ok$ ($\Pi \cup \mathcal{B} \vdash \sim fuel\_ok$). The argument $\langle \mathcal{B}, \sim fuel\_ok \rangle$ *defeats* $\langle \mathcal{A}, engine\_ok \rangle$, since it is more specific. Hence, the argument $\langle \mathcal{A}, engine\_ok \rangle$ will be provisionally rejected, since it is defeated. However, $\langle \mathcal{A}, engine\_ok \rangle$ can be *reinstated*, since there exists a third argument $\mathcal{C} = \{\sim low\_speed$ $-\!\prec$ $sw2, sw3\}$ for $\sim low\_speed$ which on its turn defeats $\langle \mathcal{B}, \sim fuel\_ok \rangle$. Note that the argument $\langle \mathcal{D}, fuel\_ok \rangle$ with $\mathcal{D} = \{fuel\_ok$ $-\!\prec$ $sw3\}$ would be also a (blocking) defeater for $\langle \mathcal{B}, \sim fuel\_ok \rangle$.

Hence, $\langle \mathcal{A}, engine\_ok \rangle$ comes to be undefeated, since the argument $\langle \mathcal{B}, \sim fuel\_ok \rangle$ was defeated. But there is another defeater for $\langle \mathcal{A}, engine\_ok \rangle$, the argument $\langle \mathcal{E}, \sim engine\_ok \rangle$, where $\mathcal{E} = \{$ *pump_fuel_ok* $-\!\prec$ *sw1*, *pump_oil_ok* $-\!\prec$ *sw2*, $fuel\_ok$ $-\!\prec$ *pump_fuel_ok*, *oil_ok* $-\!\prec$ *pump_oil_ok*, $\sim engine\_ok$ $-\!\prec$ $fuel\_ok, oil\_ok, heat\}$. Hence $\langle \mathcal{A}, engine\_ok \rangle$ is once again provisionally defeated.

The agent might try to find a defeater for $\langle \mathcal{E}, \sim engine\_ok \rangle$ which could help *reinstate* the original argument $\langle \mathcal{A}, ok \rangle$, for example $\langle \{\sim oil\_ok$ $-\!\prec$ $heat\}, \sim oil\_ok \rangle$. It must be noted, however, that this last argument would be *fallacious*, since there would exist odd-level supporting arguments for both *oil_ok* (as a subargument of $\langle \mathcal{A}, engine\_ok \rangle$) and for $\sim oil\_ok$ (in $\langle \{\sim oil\_ok$ $-\!\prec$ $heat\}, \sim oil\_ok \rangle$). Hence $\langle \{\sim oil\_ok$ $-\!\prec$ $heat\}, \sim oil\_ok \rangle$ should not be accepted as a valid defeater for $\langle \mathcal{A}, engine\_ok \rangle$. Since there are no more arguments to consider, $\langle \mathcal{A}, engine\_ok \rangle$ turns out to be ultimately defeated, so that we can conclude that the argument $\langle \mathcal{A}, engine\_ok \rangle$ is not *justified.* Thus, we conclude that the engine is not working ok. The argument $\langle \mathcal{E}, \sim engine\_ok \rangle$, on its turn, is a *justification.*

Fig. 2(b)-left shows the resulting dialectical tree. Note that $\langle \mathcal{A}, engine\_ok \rangle$ is a level-0 supporting argument, and both $\langle \mathcal{C}, \sim low\_speed \rangle$ and $\langle \mathcal{D}, fuel\_ok \rangle$ are level-2 supporting arguments. Both $\langle \mathcal{B}, \sim fuel\_ok \rangle$ and $\langle \mathcal{E}, \sim engine\_ok \rangle$ are level-1 interfering arguments. $\lambda = [\ \langle \mathcal{A}, engine\_ok \rangle, \langle \mathcal{B}, \sim fuel\_ok \rangle, \langle \mathcal{C}, \sim low\_speed \rangle\ ]$ is an *argumentation line.*

3 Pruning dialectical trees

Building a dialectical tree is computationally expensive: arguments are proof trees, and a dialectical tree is a tree of arguments. In both cases, consistency checks are needed. Thus, exhaustive search turns out to be impractical when modelling real-world situations using argumentative frameworks.

According to the definition of *justification*, a dialectical tree resembles an AND-OR tree: even though an $\langle \mathcal{A}, h \rangle$ may have many possible defeaters $\langle \mathcal{B}_1, h_1 \rangle$, $\langle \mathcal{B}_2, h_2 \rangle$, ... , $\langle \mathcal{B}_k, h_k \rangle$, it suffices to find just *one* acceptable defeater $\langle \mathcal{B}_i, h_i \rangle$ in order to consider $\langle \mathcal{A}, h \rangle$ as defeated. Therefore, when analyzing the acceptance of a given argument $\langle \mathcal{A}, h \rangle$ not every node in the dialectical tree $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ has to be expanded in order to determine the label of the root. α-β pruning can be applied to speed up the labeling procedure, as shown in figure 2(a). Non-expanded nodes are marked with an asterisk $\star$. Note: dialectical trees are assumed to be computed depth-first.

It is well-known that whenever α-β pruning can be applied, the *ordering* according to which nodes are expanded affects the size of the search space Consider our former example: when determining whether $\langle \mathcal{A}, engine\_ok \rangle$ was justified, we computed depth-first *all* arguments involved, thus obtaining the dialectical tree $\mathcal{T}_{\langle \mathcal{A}, engine\_ok \rangle}$ shown in figure 2(b)-left. However, had we started by considering the defeater $\langle \mathcal{E}, \neg engine\_ok \rangle$ before than $\langle \mathcal{B}, \neg fuel\_ok \rangle$, we would have come to the same outcome by just taking a subtree of $\mathcal{T}_{\langle \mathcal{A}, engine\_ok \rangle}$ (as shown in figure 2 (b)-right).

Computing this set exhaustively is a complex task, since we should consider *every* possible counterargument for $\langle \mathcal{A}, h \rangle$, determining whether it is an acceptable defeater or not. In order to formalize the *ordering* for expanding defeaters as the dialectical tree is being built, we will introduce a partial order $\preceq_{eval}$ as follows:

Definition 3.1. *Let S be a set of defeaters for $\langle \mathcal{A}, h \rangle$. Given two arguments $\langle \mathcal{A}_1, h_1 \rangle$ and $\langle \mathcal{A}_2, h_2 \rangle$ in S, we will say that $\langle \mathcal{A}_1, h_1 \rangle \preceq_{eval} \langle \mathcal{A}_2, h_2 \rangle$ iff $\langle \mathcal{A}_1, h_1 \rangle$'s label is computed before than $\langle \mathcal{A}_2, h_2 \rangle$'s label.*

[4] See [GSC98] for an in-depth analysis.

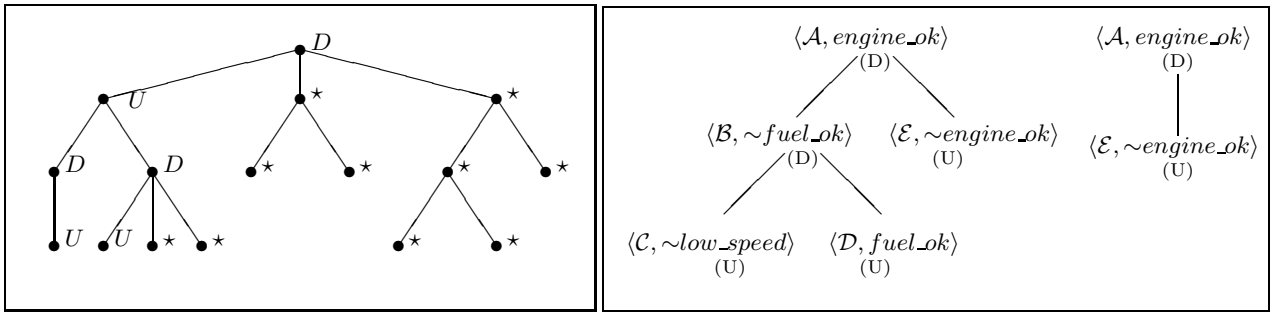

Fig. 2. (a) Labeling a dialectical tree with $\alpha - \beta$ pruning. (b) Dialectical trees (example 2.11)

Example 3.2. In example 2.11, it is the case that $\langle \mathcal{B}, \sim fuel\_ok \rangle \preceq_{eval} \langle \mathcal{E}, \sim engine\_ok \rangle$.

In dialectical trees, only *acceptable* defeaters are considered, i.e. those which are non-fallacious (as mentioned in example 2.11). Let $\langle \mathcal{A}, h \rangle$ be an argument in a dialectical tree. Then we will denote as $AcceptableDefeaters(\langle \mathcal{A}, h \rangle)$ the set $\{\langle \mathcal{B}_1, h_1 \rangle, \ldots, \langle \mathcal{B}_n, h_n \rangle\}$ of acceptable defeaters for $\langle \mathcal{A}, h \rangle$ in that tree.

Example 3.3. Consider example 2.11. It holds that $\langle \mathcal{B}, \sim fuel\_ok \rangle$ is an acceptable defeater for $\langle \mathcal{A}, engine\_ok \rangle$, whereas $\langle \{\sim oil\_ok \prec heat\}, \sim oil\_ok \rangle$ is *not* an acceptable defeater for $\langle \mathcal{E}, \sim engine\_ok \rangle$.

The algorithm in figure 3 shows how a dialectical tree can be built and labelled in a depth-first fashion, using both α-β pruning and the evaluation ordering $\preceq_{eval}$. In order to speed up the construction of a dialectical tree, our approach will be twofold. First, given an argument $\langle \mathcal{A}, h \rangle$, we will establish a syntactic criterion for determining the set $AcceptableDefeaters(\langle \mathcal{A}, h \rangle)$. Second, we will give a definition of $\preceq_{eval}$ which prunes the dialectical tree according to consistency constraints. Both approaches will be discussed in section 3.1 and 3.2, respectively.

3.1 Commitment set

We will consider three distinguished sets of literals associated with an argument $\langle \mathcal{A}, h \rangle$:
(a) the set of *points for counterargumentation* (literals which are conclusions of counterarguments for $\langle \mathcal{A}, h \rangle$);
(b) the set of *points for defeat* (literals which are conclusions of defeaters for $\langle \mathcal{A}, h \rangle$); and
(c) the set of *points for attack* (literals which are conclusions of acceptable defeaters for $\langle \mathcal{A}, h \rangle$ in a given dialectical tree).

We will denote these sets as $PointsForCounterarg(\langle \mathcal{A}, h \rangle)$, $PointsForDefeat(\langle \mathcal{A}, h \rangle)$, and $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda)$, respectively. From definitions 2.6 and 2.7, each of these sets is a subset of the preceding ones, i.e.:

$$PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda) \subseteq PointsForDefeat(\langle \mathcal{A}, h \rangle) \subseteq PointsForCounterarg(\langle \mathcal{A}, h \rangle)$$

The set $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda)$ represents the *optimal* set of literals to take into account for building defeaters for $\langle \mathcal{A}, h \rangle$, in the sense that every literal in this set accounts for a conclusion of an acceptable defeater. In [SL92], the approach to determine all possible defeaters for a given argument $\langle \mathcal{A}, h \rangle$ considered the deductive closure of the complement of the literals which are consequents of those rules (in Π and $\mathcal{A}$) used in deriving h. This notion, which will prove useful for pruning the search space, will be characterized as *commitment set*:

Definition 3.4. *Let $\mathcal{P} = (\Pi, \Delta)$ be a dlp, and let $\langle \mathcal{A}, h \rangle$ be an argument in $\mathcal{P}$. The* commitment set *of $\langle \mathcal{A}, h \rangle$ wrt $\mathcal{P}$, denoted $Commit(\langle \mathcal{A}, h \rangle)$, is defined as $Commit(\langle \mathcal{A}, h \rangle) = \{a \mid a$ is a ground literal such that $\Pi \cup \mathrm{Co}(\mathcal{A}) \vdash a\}$, where $Co(\mathcal{A})$ denotes the set of consequents of defeasible rules in $\mathcal{A}$. If $S = \{\langle \mathcal{A}_1, h_1 \rangle, \ldots \langle \mathcal{A}_n, h_n \rangle\}$ is a set of arguments, then $Commit(S) = \{\ a \mid a$ is a ground literal such that $\Pi \cup \bigcup_{i=1\ldots n} Co(\mathcal{A}_i) \vdash a\}$.*

The set $\overline{Commit(\langle \mathcal{A}, h \rangle)}$[5] is suggested in [SL92] as an approximation to $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda)$. From the preceding inclusion relationship, it follows that $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda) \subseteq \overline{Commit(\langle \mathcal{A}, h \rangle)}$. One of our goals is to find a better upper bound for $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda)$. Next we introduce a lemma to consider a proper subset of $\overline{Commit(\langle \mathcal{A}, h \rangle)}$ for finding acceptable defeaters by backward chaining, thus reducing the number of defeaters to take into account. That subset is given by the *the consequents of defeasible rules* in $\mathcal{A}$.

Lemma 3.5. [6] *Let $\langle \mathcal{A}, h \rangle$ be an argument. Let $\langle \mathcal{B}, j \rangle$ be an acceptable defeater for $\langle \mathcal{A}, h \rangle$, i.e., $\langle \mathcal{B}, j \rangle$ defeats $\langle \mathcal{A}, h \rangle$. Then B is also an argument for a ground literal q, such that q is the complement of some consequent of a defeasible rule in A, and $\langle \mathcal{B}, q \rangle$ is an acceptable defeater.*

Hence, we can find a better upper bound for the set $PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda)$ by considering the set $\overline{Co(A)}$. Note that this set can be immediately computed once the argument $\langle \mathcal{A}, h \rangle$ has been built, whereas the approach given in [SL92] involved computing the much more complex deductive closure $(\Pi \cup \mathcal{A})^{\vdash}$.

[5] $\overline{S}$ stands for the set formed by the complement of every literal in S. E.g: $\overline{\{a, \sim b\}} = \{\sim a, b\}$.
[6] The lemmas in this paper are based on the ones presented in [Che96] and [Gar97].

ALGORITHM **3.1** BuildDialecticalTree
INPUT: $\langle \mathcal{A}, h \rangle$
OUTPUT: $\mathcal{T}_{\langle \mathcal{A},h \rangle}$
{*uses* α-β *pruning and evaluation ordering* $\preceq_{eval}$ }
Let $S = AcceptableDefeaters(\langle \mathcal{A}, h \rangle)$
If $S \neq \emptyset$
  then
    While there is no $\langle \mathcal{A}_i, h_i \rangle \in S$ labelled as U
      For every argument in S
        Let $\langle \mathcal{A}_i, h_i \rangle$= minimal non-labelled element in $(S, \preceq_{eval})$
        BuildDialecticalTree($\langle \mathcal{A}_i, h_i \rangle$) getting as a result $\mathcal{T}_{\langle \mathcal{A}_i,h_i \rangle}$
        Put $\mathcal{T}_{\langle \mathcal{A}_i,h_i \rangle}$ as a immediate subtree of $\langle \mathcal{A}, h \rangle$.
    If there exists some $\mathcal{T}_{\langle \mathcal{A}_i,h_i \rangle}$ labelled as U
      then Label $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ as D
      else Label $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ as U
  else
    $\mathcal{T}_{\langle \mathcal{A},h \rangle} = \langle \mathcal{A}, h \rangle$
    Label $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ as U

Fig. 3. Algorithm for building and labelling a dialectical tree

3.2 Commitment and evaluation order. Shared basis

As remarked in section 2, fallacious argumentation is to be avoided. In *DeLP*, this means that all odd-level (even-level) arguments in an argumentation line λ= [$\langle \mathcal{A}_0, h_0 \rangle$, $\langle \mathcal{A}_1, h_1 \rangle$, ... , $\langle \mathcal{A}_k, h_k \rangle$] must be *non-contradictory* wrt Π in order to avoid *contradictory* argumentation. Def. 3.4 captures the notion of commitment set for a given argument $\langle \mathcal{A}, h \rangle$. We will use that notion for pruning the search space in order to determine possible defeaters for $\langle \mathcal{A}, h \rangle$, without considering the whole set $\overline{Co(A)}$.

Lemma 3.6. *Let λ be an argumentation line in a dialectical tree $\mathcal{T}_{\langle \mathcal{A},h \rangle}$, such that S^k_λ denotes the set of all supporting arguments in λ with level $\leq k$. Let a be a ground literal, $a \in Commit(S^k_\lambda)$. Let $\langle \mathcal{B}, j \rangle \in I_\lambda$, such that its level is greater than k. Then $\overline{a} \notin PointsForAttack(\langle \mathcal{B}, j \rangle, \lambda)$.*

This lemma establishes the following: assume that an argumentation line has been built up to level k. If an interferring argument were then introduced at level $k' > k$, it could not be further attacked by a supporting argument with conclusion $\sim a$ at level $k'' > k'$, if it is the case that a belongs to $Commit(S^k_\lambda)$. Thus, the former lemma accounts for the need of not falling into 'self-contradiction' when an argument exchange is performed. In order to introduce new supporting (interferring) arguments, the proponent (opponent) is committed to what he has stated before. This allows us to further reduce the set of literals $\overline{Co(\langle \mathcal{A}, h \rangle)}$ to take into account for determining defeaters for $\langle \mathcal{A}, h \rangle$. As an argumentation line is being built, if a is a literal in a supporting (interferring) argument at level k, its complement $\overline{a}$ cannot be the conclusion of supporting (interferring) arguments at level $k' > k$.

As a direct consequence from lemma 3.6, literals which are present in *both* supporting and interferring arguments up to level k in a given argumentation line *cannot be further argued at level* $k' > k$. We can also draw a parallel with an informal argument exchange as before: if the proponent (opponent) *concedes* some point to the opponent (proponent), that point is beyond question for further argumentation.

Definition 3.7 (SharedBasis). *Let $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ be an argumentation line in $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. We define $SharedBasis(\lambda, k)$ as the set of ground literals in the deductive closure of: a) Π; b) the consequents of rules in both S_λ and I_λ up to level k within the argumentation line λ. Formally:*[7]
$SharedBasis(\lambda, k) = \{a : a \text{ is a ground literal, and } a \in (\Pi \cup (Co(DRules(S^k_\lambda)) \cap (Co(DRules(I^k_\lambda)))^{\vdash}\}$

From this definition we can state the following lemma, which excludes literals belonging to the shared basis (up to a given level k) as points for attack for arguments at deeper levels.

Lemma 3.8 (Commitment Lemma). *Let $a \in SharedBasis(\lambda, k)$, $k \geq 0$. Then $\overline{a} \notin PointsForAttack(\langle \mathcal{B}, j \rangle, \lambda)$, for any argument $\langle \mathcal{B}, j \rangle \in \lambda$.*

From lemma 3.5 and 3.8 it follows that those literals belonging to $Co(A)$ which are in $SharedBasis(\lambda, k)$ *cannot be* the conclusions of defeaters for $\langle \mathcal{A}, h \rangle$. This allows us to get an improved upper bound for the potential points for attack when computing defeaters for a given argument $\langle \mathcal{A}, h \rangle$ at level k in a dialectical tree.

$$PointsForAttack(\langle \mathcal{A}, h \rangle, \lambda) \subseteq \overline{Co(\langle \mathcal{A}, h \rangle) - SharedBasis(\lambda, k)} \subseteq \overline{Co(\langle \mathcal{A}, h \rangle)} \subseteq \overline{Commit(\langle \mathcal{A}, h \rangle)}$$

[7] If S is a set of arguments { $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$, ... , $\langle \mathcal{A}_k, h_k \rangle$ }, then $DRules(S)$ denotes the set of all defeasible rules in S, i.e., $DRules(S) = A_1 \cup A_2 \cup \ldots \cup A_k$.

3.3 Preference criterion

From the preceding analysis we can come back to the original question: how to choose those defeaters belonging to the most 'promising' argumentation line? (i.e., those which are more prone to break the debate as soon as possible). From our preceding results, we can introduce the following definition for $\preceq_{eval}$:

Definition 3.9 (Evaluation ordering based on shared basis). *Let λ = [...., $\langle \mathcal{A}, h\rangle$] be an argumentation line whose last element is an argument $\langle \mathcal{A}, h\rangle$ at level $k-1$. Let $\langle \mathcal{A}_1, h_1\rangle$ y $\langle \mathcal{A}_2, h_2\rangle$ be two possible defeaters for $\langle \mathcal{A}, h\rangle$, so that choosing $\langle \mathcal{A}_1, h_1\rangle$ would result in an argumentation line λ_1 = [...., $\langle \mathcal{A}, h\rangle$,$\langle \mathcal{A}_1, h_1\rangle$], and choosing $\langle \mathcal{A}_2, h_2\rangle$ would result in an argumentation line λ_2 = [...., $\langle \mathcal{A}, h\rangle$,$\langle \mathcal{A}_2, h_2\rangle$]. Then $\langle \mathcal{A}_1, h_1\rangle \preceq_{eval} \langle \mathcal{A}_2, h_2\rangle$ iff*

$$\overline{Co(\langle \mathcal{A}_1, h_1\rangle) - SharedBasis(\lambda_1, k)} \subseteq \overline{Co(\langle \mathcal{A}_2, h_2\rangle) - SharedBasis(\lambda_2, k)}$$

This evaluation order can be now applied in the algorithm 3.1. An advantageous feature of this evaluation order is that it is easy to implement. Given two alternative defeaters for an argument $\langle \mathcal{A}, h\rangle$, the one which shares *as many ground literals as possible* with the argument ($\langle \mathcal{A}, h\rangle$) being attacked should be preferred. This policy maximizes naturally the set $SharedBasis$.

Example 3.10. Consider examples 2.3 and 2.11. Figure 2 showed two alternative ways of determining whether $\langle \mathcal{A}, engine\_ok\rangle$ is a justification. The consequents of defeasible rules are, in this case, $Co(\mathcal{A})$ = { $engine\_ok$, $fuel\_ok$, oil_ok, $pump\_fuel\_ok$, $pump\_oil\_ok$ }. The argument $\langle \mathcal{A}, engine\_ok\rangle$ has two acceptable defeaters: $\langle \mathcal{B}, \sim fuel\_ok\rangle$ and $\langle \mathcal{E}, \sim engine\_ok\rangle$. In the first case, $Co(\mathcal{B})$ = { $pump\_clogged$, $pump\_fuel\_ok$, low_speed }, and in the second case, $Co(\mathcal{E})$ = { $\sim engine\_ok$, $fuel\_ok$, oil_ok, $pump\_fuel\_ok$, $pump\_oil\_ok$ }. If we choose the defeater $\langle \mathcal{B}, \sim fuel\_ok\rangle$, we have $\overline{Co(A) - SharedBasis(\lambda_1, 1)}$ = { $\neg engine\_ok$, $\neg fuel\_ok$, $\neg oil\_ok$, $\neg pump\_oil\_ok$ }. Choosing the defeater $\langle \mathcal{E}, \sim engine\_ok\rangle$, we have $\overline{Co(A) - SharedBasis(\lambda_2, 1)}$ = { $\neg engine\_ok$ }. Since $\overline{Co(A) - SharedBasis(\lambda_2, 1)} \subset \overline{Cpl(Co(A) - SharedBasis(\lambda_1, 1)}$, the defeater $\langle \mathcal{E}, \sim engine\_ok\rangle$ should be tried before than $\langle \mathcal{B}, \sim fuel\_ok\rangle$ when computing the dialectical tree $\mathcal{T}_{\langle \mathcal{A}, engine\_ok\rangle}$.

4 Conclusions and related work

Defeasible Argumentation is a relatively new field in Artificial Intelligence. Inference in argument-based systems is a complex issue, and we think our results give a relevant contribution to currently existing work [PV99,Vre97]. Given two arguments $\langle \mathcal{A}_1, q_1\rangle$ and $\langle \mathcal{A}_2, q_2\rangle$, other alternative formalizations (such as Prakken and Sartor's [PV99] or Vreeswijk's [Vre97]) consider a full consistency check $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2 \vdash p, \sim p$ to determine whether two arguments attack each other. In this paper, we characterized attack in a goal-oriented way, which rendered easier many implementation issues, and helped to prune dialectical trees. It should be noted that *DeLP* [Gar97] has been implemented using this goal-oriented attack, which resembles the approach used by [KT99] for normal logic programs. However, *DeLP* provides richer knowledge representation capabilities, since it incorporates both default and strict negation.[8]

Studying the need of avoiding *fallacious* argumentation [SCG94], we arrived at the notion of *commitment* and *shared basis*, which allowed us to define a preference criterion for dynamically obtaining the (on the average) shortest argumentation lines when the justification procedure is carried out. Although our analysis was particularly focused on *DeLP*, the approach presented in this paper can be adapted to many existing argumentation systems.

References

[Che96] Carlos Iván Chesñevar. *The Problem of Inference in Argumentative Systems* (MsC Thesis). Departamento de Cs. de la Computacion, Univ. Nac. del Sur, B.Blanca, Argentina, December 1996.

[Gar97] Alejandro J. Garcia. *Defeasible Logic Programming: Definition and Implementation* (MSc Thesis). Departamento de Cs. de la Computacion, Univ. Nac. del Sur, B.Blanca, Argentina, July 1997.

[GL90] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In D. Warren and P. Szeredi, editors, *Proc. ICLP*, pages 579–597. MIT Press, 1990.

[GSC98] A. J. Garcia, G. R. Simari, and C. I. Chesñevar. An argumentative framework for reasoning with inconsistent and incomplete information. In *Workshop on Practical Reasoning and Rationality.* ECAI-98, 1998.

[KT99] Antonios C. Kakas and Francesca Toni. Computing argumentation in logic programming. *Journal of Logic and Computation*, 9(4):515–562, 1999.

[PV99] Henry Prakken and Gerard Vreeswijk. Logics for Defeasible Argumentation. In Dov Gabbay, editor, *Handbook of Philosophical Logic.* Kluwer Academic Publisher, 1999.

[SCG94] G. R. Simari, C. I. Chesñevar, and A. J. Garcia. The role of dialectics in defeasible argumentation. In *XIV Conf. Int. de la Soc. Chilena para Cs. de la Computacion.* Concepcion, Chile, November 1994.

[SL92] Guillermo R. Simari and Ronald P. Loui. A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53:125–157, 1992.

[Vre97] Gerhard A. Vreeswijk. Abstract argumentation systems. *Artificial Intelligence*, 90:225–279, 1997.

[8] A full analysis of these features is beyond the scope of this paper.