Ann Math Artif Intell (2013) 69:103–129
DOI 10.1007/s10472-013-9338-x

Using argument strength for building dialectical bonsai

Sebastian Gottifredi · Nicolás D. Rotstein · Alejandro J. García · Guillermo R. Simari

Published online: 16 March 2013

Abstract Argumentation in AI provides an inconsistency-tolerant formalism capable of establishing those pieces of knowledge that can be accepted despite having information in contradiction. Computation of accepted arguments tends to be expensive; in order to alleviate this issue, we propose a heuristics-based pruning technique over argumentation trees. Empirical testing shows that in most cases our approach answers queries much faster than the usual techniques, which prune with no guide. The heuristics is based on a measure of strength assigned to arguments. We show how to compute these strength values by providing the corresponding algorithms, which use dynamic programming techniques to reutilise previously computed trees. In addition to this, we introduce a set of postulates characterising the desired behaviour of any strength formula. We check the given measure of strength against these postulates to show that its behaviour is rational. Although the approach presented here is based on an abstract argumentation framework, the techniques are tightly connected to the dialectical process rather than to the framework itself. Thus, results can be extrapolated to other dialectical-tree-based argumentation formalisms with no additional difficulty.

This article is an extension of [25].

S. Gottifredi (✉) · N. D. Rotstein · A. J. García · G. R. Simari
National Council of Scientific and Technical Research (CONICET),
Artificial Intelligence Research & Development Laboratory (LIDIA),
Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina
e-mail: sg@cs.uns.edu.ar

N. D. Rotstein
e-mail: nicorotstein@gmail.com

A. J. García
e-mail: ajg@cs.uns.edu.ar

G. R. Simari
e-mail: grs@cs.uns.edu.ar

Keywords Non-monotonic reasoning · Computational argumentation · Dialectical proof procedures · Heuristics-based tree pruning

Mathematics Subject Classifications (2010) 68T27 · 68T37

1 Introduction & motivation

The theory on computational argumentation is usually focused on bringing new theoretical elements to augment the expressive capability of the formalism. Other extensions are also devoted to handle different aspects of the argumentation process, from its dynamics [24] to its capability to represent dialogues [23], negotiations [2], and other features [6]. Complementarily, some approaches study the suitability of argumentation within different application contexts, such as Multi-Agent Systems [27] and the Semantic Web [22]. However, many complications lying on the practical side of argumentation have not been completely addressed. Implementations have not yet achieved maturity; the few systems available are still at an experimental stage and have never been tested against large amounts of data. This is understandable for a rather young discipline like Argumentation in Artificial Intelligence.

Nonetheless, as the theoretical foundations become stronger, the community is starting to pay attention to the computational tractability of argumentation [3, 8, 10, 18]. In this article, we take on this concern and put our focus on the computation of acceptable arguments through *dialectical trees*; that is, in order to find whether an argument is *accepted*, a tree of arguments rooted in it is built and evaluated, where children of an arguments are its counterarguments. The notion of dialectical tree as a proof procedure is not new and has been studied extensively in the literature [7, 14, 17, 20]. We will refer to these accepted arguments as being *warranted*. Computing warrant in a massive argumentation scenario [21] brings about two difficulties: either dialectical trees are small and too many, or they are few but large. In this paper we address the latter issue, attempting to build smaller trees via a *pruning technique*. Thus, a smaller amount of arguments would be required to determine the status of the root argument. In order to confirm this improvement, experimental tests were performed over large argumentation frameworks, randomly generated, containing up to 500 arguments. Empirical results are given in Section 5.

In this article we apply the pruning technique over dialectical trees taken from a variation of Dung's abstract framework for argumentation [16] called *dynamic argumentation framework* that considers a *universal* set of arguments along with a subset of currently *active* ones. From these two sets of arguments we can consider two kinds of trees: *potential* and *active*; the former refers to trees built using the universal set of arguments, whereas the latter is restricted to the set of active arguments. Only active trees are used to compute warrant, as they represent the current state of the world. As will be clear later, potential trees are used for calculating each argument's heuristic value. Consider the dialectical tree depicted in Fig. 1a. Assume this is the potential tree for argument A. Two possible active instances are shown in (b) (H and D are inactive, and possibly K) and (c) (D, E, G, H and I are inactive, and possibly J and K). These active trees would be the ones used by the system to compute warrant by exploring their nodes and "labelling" the root node as warranted or not. In order to minimise the amount of nodes explored, we introduce a pruning

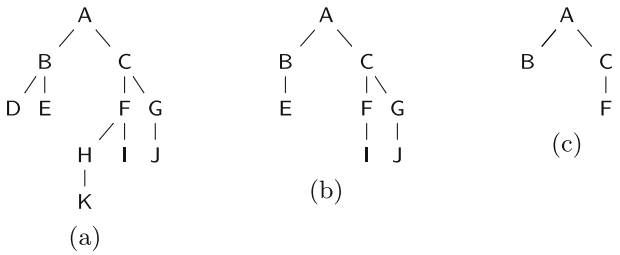


Fig. 1 Potential tree and two possible active trees

technique. Our approach looks for maximising the opportunities for pruning active trees based on the information obtained from the corresponding potential trees, regardless of how different these are—e.g., the active tree in Fig. 1c wrt. the potential tree in Fig. 1a. Unlike preexisting articles on pruning of dialectical trees, we provide a full, generalised formalisation and then analyse its implications in terms of the performance gain for argumentative query answering.

The approach used for pruning will be based on the abstract notion of *argument strength*, which indicates the likelihood of an argument to be ultimately defeated, as described in Section 3.2. We propose a concrete formula, although others could be considered. Arguments' strength is used as a heuristic value to sort the attackers of an inner node during the construction of dialectical trees. We will show that pruning opportunities are likely to appear, and entire subtrees can be omitted without affecting the calculation of the root's warrant. We introduce a strategy to build these *dialectical bonsai*,[1] based on our proposal of argument strength. Besides this particular formula, we also present a set of postulates looking to characterise any strength function. Then, we check the formula against the postulates to verify its sensibility.

In order to validate our approach, in Section 5 we perform a series of tests. Empirical testing shows that the usage of strength values allows for virtually instantaneous response to queries. In parallel with strength calculation we can also store arguments and attacks, saving the building stage every time a query takes place. Computing a strength value for each argument comes at a certain cost. In order to alleviate this, our algorithm relies on dynamic programming by reusing large portions of data. In Section 3.2 we present the algorithm corresponding to strength calculation; empirical analysis shows that this computation does not undermine the gain obtained by bonsai, in terms of time.

Although the goal of this paper is mainly practical, here we study the suitability of the approach applied over a particular flavour of Dung's framework [16]. Concepts and definitions are grounded on argumentation mechanisms rather than properties of this abstract framework. Therefore, the approach can be translated in terms of concrete argumentation frameworks rather straightforwardly. For instance, think of a rule-based argumentation formalism. Provided that there are no functional letters, we could build the universal set of arguments using a subset of the (finite) *Herbrand Base*. Then, as the state of the world changes, facts would be asserted and retracted

[1] The pruning technique attempts to keep trees small while retaining the properties of the entire tree, like a bonsai.

accordingly, therefore activating and deactivating arguments. Here we will show how to compute a heuristics from the universal set of arguments in order for it to be useful at any given state of the world. We contend that many implementations for argument frameworks would benefit from these results.

The paper is organised as follows: Section 2 provides the main theoretical elements needed for the rest of the article. Section 3 explains what is the notion of strength used by our approach, presenting postulates and a concrete formula. Section 4 describes and defines the concept of dialectical bonsai and its relation to the usual notion of pruning. Section 5 presents the empirical tests that justify the usage of our heuristics to achieve a more efficient pruning. Sections 7 and 8 discuss related and future work. Finally, Section 6 outlines the paper and concludes.

2 Theoretical basis

Nowadays, Dung's abstract argumentation framework has become the standard to analyse and apply new ideas to any argumentation-based setting. In this article, we will use a more general version of this classic framework that allows for the representation of both *active* and *inactive* arguments. This *dynamic abstract argumentation framework* (DAF) is a simpler version than the one presented in [24], getting rid of all the representational intricacies that are not relevant for this line of research. This version of the DAF consists of a universal set of arguments holding every conceivable argument along with a subset of it containing only the active ones. These active arguments represent the current state of the world and are the only ones that can be used by the argumentation machinery to make inferences and compute warrant. Therefore, at a given moment, those arguments from the universal set that are not active represent reasons that, though valid, cannot be taken into consideration due to the current context. At some point, active arguments could become inactive, thus no longer used for inference, or *vice versa*. Having inactive arguments is the very reason behind using the DAF: it will allow us to evaluate potential dialectical trees in order to obtain arguments' heuristic values that are afterwards used to prune active trees. Furthermore, one of the scenarios that would find our pruning technique advantageous is that of an argumentation-based agent. Continuous querying requires an almost instantaneous response. Imagine an agent looking for warrants ten times a second (which is a rather conservative approximation). If each warrant takes a tenth of a second, the agent would either lose half its perceptions or yield constantly delayed conclusions about what to do next. It will be clear by the end of Section 5 that our approach requires a time for precompilation to then have the capability of delivering almost instantaneous query answering.

Definition 1 (Dynamic argumentation framework) A **dynamic argumentation framework**, or DAF, is a triple $(\mathbb{U}, \hookrightarrow)[\mathbb{A}]$, where $\mathbb{U}$ is the universal set of arguments, $\hookrightarrow \subseteq \mathbb{U} \times \mathbb{U}$ is the attack relation between arguments, and $\mathbb{A} \subseteq \mathbb{U}$ is the subset of active arguments.

The DAF yields a graph of arguments connected by the attack relation. An "active subgraph" could be considered, containing only active arguments. In argumentation, the challenge consists in finding out which active arguments prevail after all things

considered, i.e., those arguments that are warranted. To this end, the notion of *argumentation semantics* has been extensively studied [4]. In this article, warrant of arguments will be determined on top of the dialectical tree for each one of them, assuming a particular marking criterion (see Assumption 1).

A dialectical tree is conformed by a set of argumentation lines; each of which is a non-empty sequence λ of arguments from a DAF, where each argument in λ attacks its predecessor in the line. An argumentation line should be non-circular (an argument should not occur twice in the same argumentation line) in order to avoid infinite lines, and it should be also exhaustive, i.e., no more arguments can be added to it.

Definition 2 (Argumentation line) Given a DAF $\tau = (\mathbb{U}, \hookrightarrow)[\mathbb{A}]$, and $\mathsf{B}_1, \ldots, \mathsf{B}_n \in \mathbb{U}$, an **argumentation line** λ in τ is a (non-empty) finite sequence of arguments $[\mathsf{B}_1, \ldots, \mathsf{B}_n]$ such that, $\forall \mathsf{B}_i, \mathsf{B}_j$ with $i \neq j$ and $1 < i, j \leq n$, $\mathsf{B}_i \hookrightarrow \mathsf{B}_{i-1}$, $\mathsf{B}_i \neq \mathsf{B}_j$ and $\nexists \mathsf{C} \in \mathbb{U}$ such that $\mathsf{C} \hookrightarrow \mathsf{B}_n$. The argumentation line λ is said to be rooted in B_1. The set of all argumentation lines in τ is noted as $\mathfrak{Lines}_\tau$.

The first argument in an argumentation line λ is called the *root* whereas the last one is the *leaf* of λ. Arguments in these lines are classified according to their role wrt. the root argument: a *pro argument* (respectively, *con*) in an argumentation line is placed at an odd (respectively, even) position. This classification can be extended to any argument in the tree; positions must be interpreted relative to the argument at issue. For instance, for an argument in an even position in a line, pro arguments will be in even positions in that line.

Note that the definition for an argumentation line takes into account every argument in the universal set. A restricted version of argumentation line could be considered, setting its domain within the set of active arguments. This variant is called *active argumentation line*. Since regular argumentation lines as defined above include both active and inactive arguments, we will refer to them as *potential argumentation lines*, to emphasise their meaning.

As said before, the warrant status of an argument will be determined by analysing the dialectical tree rooted in it. A dialectical tree rooted in an argument A will be built from a set of argumentation lines rooted in A.

Definition 3 (Dialectical tree) Given a DAF $\tau = (\mathbb{U}, \hookrightarrow)[\mathbb{A}]$ and an argument $\mathsf{A} \in \mathbb{U}$, the **dialectical tree** $\mathcal{T}(\mathsf{A})$ rooted in A from τ is built from the set $X \subseteq \mathfrak{Lines}_\tau$ of every argumentation line rooted in A. The set of all dialectical trees in τ is noted as $\mathfrak{Trees}_\tau$.

As a convention, an argument C in a dialectical tree $\mathcal{T}(\mathsf{A})$ built from a set of argumentation lines X is denoted as:

- a **node** iff C appears in $\lambda \in X$
- a **child** of a node B in $\mathcal{T}(\mathsf{A})$ in λ iff $\lambda = [\ldots, \mathsf{B}, \mathsf{C}, \ldots], \lambda \in X$
- a **leaf** of $\mathcal{T}(\mathsf{A})$ in λ iff C is a leaf in $\lambda \in X$

Dialectical trees are built from argumentation lines and can be classified in a similar way. Therefore, *potential dialectical trees* are built from potential argumentation lines: those containing arguments from the universal set. Similarly, *active dialectical*

trees are built from active argumentation lines, which include only active arguments. Note that an argument may appear several times in a dialectical tree, each time in a different argumentation line. Also, given a DAF $(\mathbb{U}, \hookrightarrow)[\mathbb{A}]$, for each argument B in $\mathbb{U}$ there will be a unique potential dialectical tree that is rooted in B, and for each A in $\mathbb{A}$ there will be a unique active dialectical tree rooted in A.

The computation of warrant through dialectical trees usually relies on a marking criterion that could be defined according to any conceivable policy; its main objective is to assign a status to each argument in the dialectical tree. The status of the root argument would tell whether it is warranted. An abstract specification for a sensible marking criterion was given in [26]. Here we present a particular version of the *marking function* that assigns either "**D**" (defeated) or "**U**" (undefeated) to each argument.

Definition 4 (Marking function) Given a DAF τ, a **marking function** over arguments in τ is any function $\mathfrak{m} : \mathbb{U} \times \mathfrak{Lines}_\tau \times \mathfrak{Trees}_\tau \rightarrow \{\mathbf{D}, \mathbf{U}\}$.

Since the same argument can appear in different lines of the same tree, the marking function needs to address it through line and tree, e.g., an argument could be marked as **D** in some lines and **U** in others. Although Definition 4 indicates that $\mathfrak{m}$ is defined for the whole cartesian product of lines, trees and arguments in a DAF, an implementation would probably define the function only for arguments within a given line within a given dialectical tree. There is a case in which an argument B can be associated to several lines, interchangeably: when the path from the root to B coincides in these lines. We will not address this issue with any particular convention, since it will not be problematic: the mark of B in all these lines will be just the same (see Example 1). In this article we assume the marking criterion given in DeLP [17].

Assumption 1 *Given a dialectical tree, a node is marked* **D** *iff it has a child marked* **U***; otherwise it is marked as* **U**.

It is worthwhile mentioning that this marking criterion yields an argumentation semantics similar to Dung's grounded semantics. Section 7 provides a deeper explanation for the relation between the two approaches.

Example 1 Consider the dialectical tree $\mathcal{T}(\mathsf{A})$ in Fig. 2, depicted along with its two argumentation lines λ_1 and λ_2. White (black) triangles are used to denote arguments marked as undefeated (defeated).

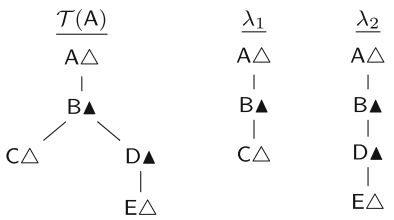


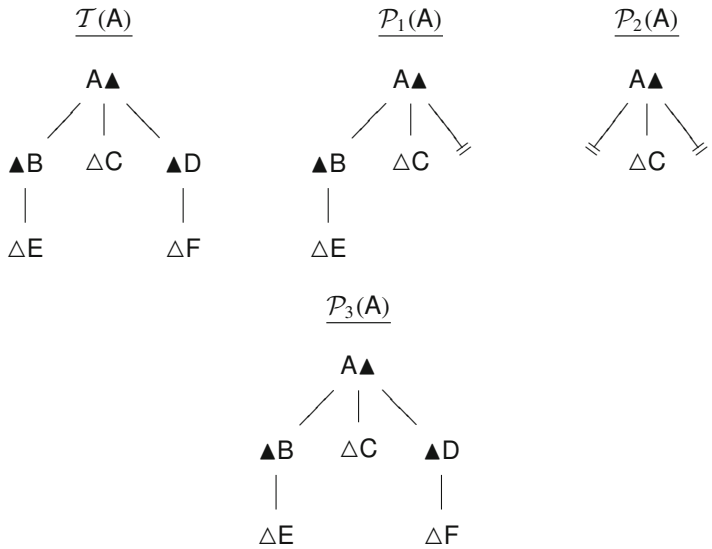
Fig. 2 The dialectial tree of Example 1 and two of its argumentation lines

Leaves are undefeated and B is defeated due to C being undefeated. Also note that A and B belong to both lines, thus receiving the same mark. Finally, pro arguments for A would be A itself plus C and D, whereas B and E would be con.

The construction of *dialectical bonsai* depends on discovering opportunities to prune dialectical trees. Given a tree $\mathcal{T}$, a bonsai of $\mathcal{T}$ will be a pruned version of it such that the marking applied to the bonsai retains, at least, the same marking for the root than the one in $\mathcal{T}$. That is, during the construction of dialectical trees, it should be possible to determine when the construction procedure has collected enough information to compute the same mark for the root as the one in the non-pruned tree. A simple example is a scenario in which there is an undefeated attacker for the root; when such an argument is found, the dialectical analysis should be stopped and the root marked as defeated. To simplify things, we will assume the *and-or* pruning technique explained in [15], which we call *1U pruning* (formally introduced in Definition 6): *whenever an* ***attacker*** *for an inner argument* A *is found as* ***undefeated****,* A *can be directly marked as* ***defeated*** *and the rest of the attackers for* A *can be ignored.*

Although this technique is not too restrictive and could even be used quite often, finding all the opportunities for pruning (thus obtaining the smallest possible trees) requires just plain luck, since undefeated attackers must be found first (see Example 2), and undefeatedness is not predictable –indeed, its discovery is the reason to build dialectical trees.

Example 2 Consider the dialectical tree $\mathcal{T}(\mathsf{A})$ depicted below, and three possible prunings: $\mathcal{P}_i(\mathsf{A})$, $1 \leq i \leq 3$. Depending on the order in which attackers for A are selected, different prunings come up.



- $\mathcal{P}_1(\mathsf{A})$: C is selected after B but before D, so D is cut off;
- $\mathcal{P}_2(\mathsf{A})$: C is selected first, both B and D are cut off;
- $\mathcal{P}_3(\mathsf{A})$: C is selected last, no pruning occurs.

Note that pruning out argument C would lead to a *faux* pruning, marking the root as undefeated. Such a pruning should and will not qualify as a bonsai.

Next, we formalise the generalised notion of pruning, according to the intuitions previously presented. This definition is based on active dialectical trees, as they are the ones used to compute warrant. A definition for pruning over potential trees would be aimed at making the strength calculation more efficient; this topic will not be discussed in this article.

Definition 5 (Pruning) Let $\mathcal{T}(\mathsf{A})$ be an active dialectical tree in the context of a DAF $\tau = (\mathbb{U}, \hookrightarrow)[\mathbb{A}]$, $\mathsf{S_t} \subseteq \mathbb{U}$, the set of arguments in $\mathcal{T}(\mathsf{A})$, and $\mathsf{E_t} \subseteq \hookrightarrow$, the set of edges in $\mathcal{T}(\mathsf{A})$. A **pruning** $\mathcal{P}(\mathsf{A})$ for $\mathcal{T}(\mathsf{A})$ is a tree rooted in A with a set of arguments $\mathsf{S_p} \subseteq \mathsf{S_t}$ and a set of edges $\mathsf{E_p} \subseteq \mathsf{E_t}$.

The above definition gives a general notion of what we consider as a pruning. However, not every pruning of a tree qualifies as a dialectical bonsai. As stated before, the requirement for a pruning to be a bonsai is to yield the same information than the complete tree about the warrant status of the root argument. Next, we introduce the particular kind of pruning our approach is based on. The reader should know that this is not a new concept, but used in existing argumentation systems, such as DeLP [17].

Definition 6 (1U Pruning) Given a DAF τ and a pruning $\mathcal{P}(\mathsf{A})$ for a dialectical tree $\mathcal{T}(\mathsf{A})$, let B be an inner node in $\mathcal{P}(\mathsf{A})$ with a set of attackers Γ in $\mathcal{T}(\mathsf{A})$ such that the subset of attackers in $\mathcal{P}(\mathsf{A})$ is $\Gamma' \subseteq \Gamma$. The pruning $\mathcal{P}(\mathsf{A})$ is a **1U pruning** for $\mathcal{T}(\mathsf{A})$ iff $\exists \mathsf{B}_i \in \Gamma, \mathfrak{m}(\mathsf{B}_i, \lambda_i, \mathcal{T}(\mathsf{A})) = \mathbf{U}$ implies that there is exactly one argument $\mathsf{B}_k \in \Gamma', \mathfrak{m}(\mathsf{B}_k, \lambda_k, \mathcal{P}(\mathsf{A})) = \mathbf{U}$.

In words, a 1U pruning is a pruning $\mathcal{P}(\mathsf{A})$ such that for any set of attackers with at least one attacker marked as **U** in the original tree $\mathcal{T}(\mathsf{A})$, the subset of attackers that stays in $\mathcal{P}(\mathsf{A})$ has exactly one undefeated attacker. Recall that by Assumption 1 the marking of a node will be **D** iff it has at least one attacked marked **U**. Therefore, the 1U pruning has only one **U** attacker since it is enough to determine the marking of the attacked argument. The definition does not specify how to treat a set Γ with all defeated arguments. Note that any subset would work, even the empty set. However, an implementation would most likely check all the defeaters to find out that they are defeated. In Example 2 all prunings qualify as a 1U pruning.

Generally in the literature, when using a pruning technique, the expansion of attackers (i.e., children) while building trees follows no criterion, and this happens both in theoretical approaches as well as in implementations. The former usually present a set of arguments to choose from, whereas the latter are often rule-based, and rules are placed rather arbitrarily, and looked up in a top-to-bottom fashion. To sum up, when building dialectical trees there is no available information to know beforehand how to augment the possibilities of pruning. An external mechanism should provide such knowledge. To this end, next we introduce the concept of argument strength.

3 Strength of an argument

Our approach to the notion of *argument strength* is similar to the heuristics proposed for argument gradual valuation [7, 11, 19], and it is based on the following statement: *"an argument is as strong as weak are its attackers"*. In this way, the number of attackers is not the only parameter that affects an argument's strength. This is particularly interesting, since a strategy would be flawed if based solely on that number. For instance, given an argument K within a dialectical tree, a subtree rooted in K packed with pro arguments should give K a high strength value. The strength of an argument in a tree should somehow codify how likely is this argument to be ultimately un/defeated. An argument with great strength could even be considered as a leaf, which would improve pruning at the risk of losing soundness. Strength values will be used by a heuristic method to prune the tree, as will be explained later. The idea behind this strength measure is to codify the likeliness of an argument to be ultimately defeated.

3.1 Postulates for an argument strength formula

The strength of an argument A is calculated from the potential dialectical tree $\mathcal{T}$ rooted in A. In this article we provide a way to extract the strength of an argument out of its associated potential tree; therefore, when a particular situation is analysed and the active tree is built, the strength of arguments will provide a heuristics to guide its construction and early discovery of pruning opportunities. Not any formula would serve to the purpose of accelerating warrant. In what follows, we introduce some postulates that would rule the proper behaviour of any strength-measuring function to be used as a heuristics. This implies constraining the way in which the strength value is propagated throughout the tree, and determining the particular contexts in which an argument's strength should be higher than another's. It is important to keep in mind that the relative strength of non-root arguments within a potential tree refers to their local measures of strength, since these are the values used to finally determine the root's actual strength. Below are the intuitions behind a collection of desirable postulates for an arbitrary strength measure function:

(**S1**) **immediate weakening:** *"An argument gets weaker as its attackers get stronger"*. This captures the initial intuition for the notion of strength, in which the strength of the set of attackers has an inverse correlation to the strength of the argument under attack.

(**S2**) **no-defeat strength:** *"Non-attacked arguments are the strongest ones"*. It is not sensible to admit that an unattacked argument could be weaker than one that is not. Although this principle suggests that leaves must have maximum strength, it does not invalidate the fact that inner nodes could also have maximum strength.

(**S3**) **strengthening by support:** *"Extra pro arguments or less con arguments increases strength"*. Adding a new pro (or removing a con) argument to a subtree increases the strength of its root. A change in strength always has a discernible cause.

(**S4**) **weakening by opposition:** *"Extra con arguments or less pro arguments decreases strength"*. Analogous to (S3).

(**S5**) **ancestral strengthening:** *"The increase in strength of an argument increases the strength of all the pro ancestors while decreasing the strength of all con"*. A change in a subtree provoking an increase in its root strength implies an increase to each of the root's pro ancestors.

(**S6**) **ancestral weakening:** *"The decrease in strength of an argument decreases the strength of all the pro ancestors while increasing the strength of all con"*. Analogous to (S5).

The changes mentioned by postulates (**S3**) to (**S6**) do not imply that trees will or have to be changed. These changes rather reveal the links between arguments and the behaviour of pro arguments wrt. the presence of cons, and *vice versa*.

It is also important to note that these postulates comply with the intuitions behind the four principles for the heuristics used in the gradual valuation for Dung's argumentation frameworks presented in [11]. In particular, postulates (**S1**), (**S2**), (**S3**) and (**S4**) can be seen as adaptations of the four principles in terms of the structure of the dialectical trees. Meanwhile, (**S5**) and (**S6**) capture new intuitions arising from the dialectical nature of these trees.

Next, the collection of postulates is listed, considering a DAF τ, two potential trees $\mathcal{T}_1$ and $\mathcal{T}_2$ from τ, and two arguments A_1 and A_2 placed in an arbitrary position in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. We also assume that an argument A in an argumentation line λ in a tree $\mathcal{T}$ has a measure of strength denoted $\mu(\mathsf{A}, \lambda, \mathcal{T})$ and an associated set of attackers $\mathsf{atts}(\mathsf{A}, \lambda, \mathcal{T})$ in $\mathcal{T}$.[2] The postulates assume that the strength function $\mu(\cdot, \cdot, \cdot)$ returns a real number within the interval $[0, 1]$. This will simplify comparison among measures of strength. Finally, the conjoint measure of strength of a set of attackers for A $\mu_{\mathsf{atts}}(\mathsf{A}, \lambda, \mathcal{T}) = \sum(\mu(\mathsf{B}_i, \lambda_i, \mathcal{T}))$, where $\mathsf{B}_i \in \mathsf{atts}(\mathsf{A}, \lambda, \mathcal{T})$.

In the case of (S3) to (S6) we consider a potential tree $\mathcal{T}_1^{\Delta}$ that is the result of arbitrarily adding and/or subtracting arguments to the subtree of $\mathcal{T}_1$ for which A_1 is the root. The notation $\textsc{pro}(\mathsf{A}, \lambda, \mathcal{T})$ (resp., $\textsc{con}(\mathsf{A}, \lambda, \mathcal{T})$) returns the set of pro (resp., con) arguments in the subtree of $\mathcal{T}$ rooted in the argument A from a line λ.

(**S1**) $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2)$ iff $\mu_{\mathsf{atts}}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \leq \mu_{\mathsf{atts}}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$

(**S2**) if $\mathsf{atts}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) = \emptyset$ and $\mathsf{atts}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \neq \emptyset$ then $\mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) < \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$

(**S3**) if $|\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \leq |\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ and $|\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \geq |\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ then $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$

(**S4**) if $|\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \leq |\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ and $|\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \geq |\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ then $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$

(**S5**) if $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ then $(\forall \mathsf{B} : \mathsf{A}_1 \in \textsc{pro}(\mathsf{B}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1^{\Delta})$ and $(\forall \mathsf{C} : \mathsf{A}_1 \in \textsc{con}(\mathsf{C}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1^{\Delta})$

(**S6**) if $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ then $(\forall \mathsf{B} : \mathsf{A}_1 \in \textsc{pro}(\mathsf{B}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1^{\Delta})$ and $(\forall \mathsf{C} : \mathsf{A}_1 \in \textsc{pro}(\mathsf{C}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1^{\Delta})$

Any formula used to calculate argument strength should look to satisfy these rationality postulates. In the following subsection we present a concrete strength formula to then check it against these postulates.

[2] Not every attacker of A will be a child in $\mathcal{T}$.

3.2 An approach to argument strength

Here we propose a formula to calculate strength for an argument A that works over the potential tree rooted in A. The same formula, following the same intuition but used for other purposes, is defined in [7]. In order to get A's strength, the method relies on the strength of A's immediate attackers, which in turn rely on the strength of their own immediate attackers, and so on, thus leading to the consideration of the entire tree. Similarly, other formulas showing the desired behaviour could be proposed.

Definition 7 (Argument strength) Let B be an argument in a line λ in a potential tree $\mathcal{T}(\mathsf{A})$, and X is a set of arguments such that every $\mathsf{C}_i \in X$ is a child of B in a line λ_i within $\mathcal{T}(\mathsf{A})$. The **strength** B in λ in the potential tree $\mathcal{T}(\mathsf{A})$ is calculated as:

$$\mu(\mathsf{B}, \lambda, \mathcal{T}(\mathsf{A})) = \begin{cases} 1, & X = \emptyset \\ \dfrac{1}{1 + \sum_i (\mu(\mathsf{C}_i, \lambda_i, \mathcal{T}(\mathsf{A})))}, & X \neq \emptyset \end{cases}$$

Again, since an argument may appear several times in the same tree but in different lines, it has to be individualised through these three parameters. The strength of an argument A in the context of a DAF τ is calculated as $\mu(\mathsf{A}, \cdot, \mathcal{T}(\mathsf{A}))$, and the shortcut is $\mu(\mathsf{A})$. Note that this formula captures the intuition that an argument's strength has an inverse correlation with the strength of its attackers.

Each argument's strength is calculated by building the potential dialectical tree rooted in it. Although each non-root argument in this potential tree has an associated strength, this measure is *local*. That is, this is not necessarily the *actual* strength value they would have in the potential tree rooted in them, but just a partial measure towards the calculation of the root's strength. Moreover, an argument appearing twice in a tree would probably have two different local strength measures. Nonetheless, the strength value associated to an argument will be its actual strength value.

Example 3 Suppose a DAF $(\{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}, \mathsf{E}\}, \{(\mathsf{B}, \mathsf{A}), (\mathsf{C}, \mathsf{A}), (\mathsf{A}, \mathsf{C}), (\mathsf{D}, \mathsf{C}), (\mathsf{C}, \mathsf{E})\})[\mathbb{A}]$. The set of active arguments here is not important, as we are looking to compute strength. The potential trees for $\mathcal{T}(\mathsf{A})$, $\mathcal{T}(\mathsf{C})$ and $\mathcal{T}(\mathsf{E})$ are shown in Fig. 3. In the tree for A, arguments B and D receive no attacks, thus $\mu(\mathsf{B}, \lambda_1, \mathcal{T}(\mathsf{A})) = 1 = \mu(\mathsf{D}, \lambda_2, \mathcal{T}(\mathsf{A}))$. Argument C is attacked just by D, then $\mu(\mathsf{C}, \lambda_2, \mathcal{T}(\mathsf{A})) = 1/(1+1) = 0.5$. Finally, A is attacked by B and C, which sum 1.5,

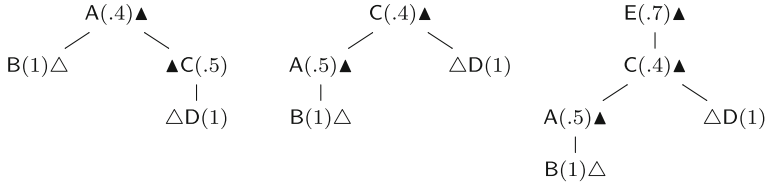


Fig. 3 Potential trees for A,C and E from DAF of Example 3 and its respective strength measures

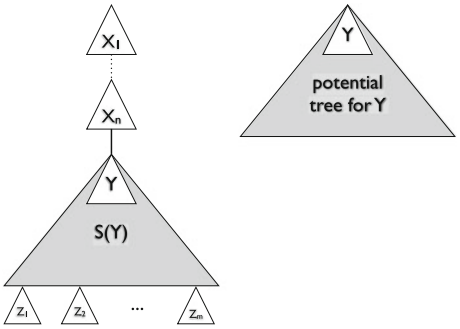


Fig. 4 Reutilising subtrees

and therefore $\mu(\mathsf{A}) = 1/(1 + 1.5) = 0.4$. The rest of the strength values are local. The actual strength of C is $\mu(\mathsf{C}) = 1/(1 + 1.5) = 0.4$, since in its potential tree $\mathcal{T}(\mathsf{C})$ has D as undefeated attacker with $\mu(\mathsf{D}, \lambda_1, \mathcal{T}(\mathsf{C})) = 1$ and A as defeated attacker with $\mu(\mathsf{A}, \lambda_2, \mathcal{T}(\mathsf{C})) = 1/(1 + 1) = 0.5$. Note that the local strength value 0.5 for C in the potential tree for A differs from C's actual strength value 0.4, computed from its potential tree. Our approach would use the actual values (rather than the local ones) in order to perform pruning, i.e., $\mu(\mathsf{A}) = \mu(\mathsf{C}) = 0.4$. Note that the local strength for C in the potential tree for E is exactly its actual strength. This is not by chance, but because the potential tree for C is a subtree there.

Having the potential tree for C as a subtree in E's is a hint for reutilising subtrees, thus making strength computation more efficient. Next, we will present an algorithm for calculating a DAF's arguments actual strength that involves this improvement by detecting the circumstances under which actual and local strength for an argument coincide, i.e., when a potential tree is completely included in another argument's tree. By looking at Fig. 4 we have the potential tree $\mathcal{T}(\mathsf{X}_1)$ for X_1 containing a subtree $\mathcal{S}(\mathsf{Y})$ rooted in Y. Arguments Z_1 to Z_m denote those that are not part of $\mathcal{T}(\mathsf{X}_1)$ due to the definition for an argumentation line prohibiting arguments to appear twice. The challenge is to be able to tell when $\mathcal{S}(\mathsf{Y})$ is exactly the potential tree for Y. In order to detect this we have to check whether arguments X_1 to X_n were left out of $\mathcal{S}(\mathsf{Y})$. Therefore, if $\{\mathsf{X}_1, \ldots, \mathsf{X}_n\} \cap \{\mathsf{Z}_1, \ldots, \mathsf{Z}_m\} = \emptyset$ then $\mathcal{S}(\mathsf{Y})$ is the potential tree for Y. In this way, when building $\mathcal{T}(\mathsf{X}_1)$ we can also store the potential tree for Y.

The symmetrical case takes place when we already have Y's potential tree calculated. When building $\mathcal{T}(\mathsf{X}_1)$, once Y is reached, we can check whether arguments X_1 to X_n belong to Y's potential tree. If they do not, then we can use Y's actual strength value, as it is the same as the local value, avoiding the construction of $\mathcal{S}(\mathsf{Y})$.

Algorithm 1 allStrengths

```
In    : a DAF (𝕌, ↪)[𝔸]
Out   : a set StrSet containing the actual strength of every argument in 𝕌
begin
    StrSet ← ∅
    foreach argument A in 𝕌 do
        getStrength(A, ∅, (𝕌, ↪)[𝔸], AStr, ATreeArgs, Excluded, StrSet)
        StrSet ← StrSet ∪ {(A, AStr, ATreeArgs)}
```

The algorithm `allStrengths` will return the actual strength of every argument in a given DAF. It will use the algorithm `getStrength` which computes the actual strength for any argument whose strength is not already calculated. Once the actual strength of an argument A is obtained, it is added to the set *StrSet* along with the arguments used to build the potential tree for A. Arguments within the potential tree will be used by the `getStrength` algorithm to determine if A's actual strength can be reutilised.

The `getStrength` algorithm is also used to compute arguments' local strength. It receives an argument A from a DAF $(\mathbb{U}, \hookrightarrow)[\mathbb{A}]$ and an argumentation line L in which A will be added, and returns the local strength of A, the arguments considered in A's subtree used to compute the local strength and the arguments excluded from it. In addition, this algorithm uses the set *StrSet* containing the actual strengths already calculated.

Algorithm 2 getStrength

In : an argument A, an argumentation line L, a DAF $(\mathbb{U}, \hookrightarrow)[\mathbb{A}]$
Out : the strength value $AStr$ of A in the context of L, the set $ATreeArgs$ containing the arguments used to calculate $AStr$, and the set $ExcludedA$ of arguments excluded when obtaining $AStr$
In-Out: the set $StrSet$ of already calculated strength values

```
begin
    attsStr ← 0
    ExcludedA ← ∅
    foreach attacker B of A in ↪ do
        if B is in L then
            ExcludedA ← ExcludedA ∪ {B}
        else
            if B is in StrSet and no argument for B in StrSet is in L then
                BStr ← getStr(B, StrSet)
                BTreeArgs ← getTreeArgs(B, StrSet)
                ExcludedB ← ∅
            else
                getStrength(B,L ∪ {A},(𝕌, ↪)[𝔸],BStr,BTreeArgs,ExcludedB,StrSet)
            attsStr ← attsStr + BStr
            ATreeArgs ← ATreeArgs ∪ BTreeArgs
            ExcludedA ← ExcludedA ∪ ExcludedB
    AStr ← 1/(1 + attsStr)
    if no Argument in ExcludedA is in L then
        StrSet = StrSet ∪ {(A, AStr, ATreeArgs)}
```

Given an argument A, `getStrength` analyses every possible attacker B of A in the DAF. If B is already in the argumentation line L, then it cannot be added to A's subtree and therefore it is excluded (it will not affect A's local strength). Otherwise, B will be part of A's subtree. In this case, if B's actual strength is already calculated and no argument in B's potential tree is part of line L, then B's actual strength stored in *StrSet* will be B's local strength in A's subtree. In particular, all B's potential tree arguments stored in *StrSet* will be part of A's subtree. On the contrary, if B's actual strength was not previously calculated, then it is obtained by recursively calling this algorithm in the context of A's subtree. Similarly, if any B's potential tree argument cannot be included in A's subtree, then it is necessary to compute B's local strength. After B's strength is obtained it is added to the sum of A's attackers strength, and the arguments used to compute B's local strength are included into A's subtree. Then, after all A's attackers have been considered, A's local strength is obtained by using

the formula presented in Definition 7. If none of the excluded arguments from A's subtree appeared previously on Line, then A's subtree is indeed A's potential tree. Therefore, A's local strength is in particular A's actual strength and can be stored in *StrSet*.

It is important to note that, when the `getStrength` algorithm receives an empty argumentation line for an argument A, it will be calculating A's actual strength directly. In particular, this is how the `allStrengths` algorithm calculates the actual strength of arguments.

Now that we have given the strength formula and presented the algorithms that implements it, we will show that the formula for calculating strength presented in Definition 7 satisfies the postulates from Section 3.1.

Proposition 1 *The strength measure* $\mu(\cdot)$ *from Definition* 7 *satisfies the postulates* (***S1***)*,* (***S2***)*,* (***S3***)*,* (***S4***)*,* (***S5***) *and* (***S6***)*.*

Proof See Appendix. □

4 Building dialectical bonsai

In this article, we will use strength values to devise a heuristics-based method to construct *dialectical bonsai*. Remember that the strength of an argument is computed on top of the potential dialectical tree associated to it. Therefore, these values will be an indication of how likely to be defeated an argument is, but given a specific scenario the real strength of an argument (corresponding to the active tree rooted in it) could differ greatly from the one calculated from the potential tree. That is, if strength values were to be kept up to date, they should be recalculated every time the situation changes, which is rather undesirable. In this article we propose a more pragmatic approach, in which each strength value is computed just once from the corresponding potential dialectical tree. Thus, when faced to particular situations, strength values end up being approximated; however they are still useful as a heuristics, as demonstrated by experimental testing.

Example 4 Consider Example 3, where $\mu(\mathsf{A}) = 0.4, \mu(\mathsf{B}) = \mu(\mathsf{D}) = 1, \mu(\mathsf{C}) = 0.4, \mu(\mathsf{E}) = 0.7$. Let us assume a world in which A and D are inactive. In this case, C has no defeaters, but its strength value is still 0.4, as it would have been calculated beforehand, from its potential tree. For instance, if while building a tree we encounter that the set of attackers is {B, C, D, E} then, given their strength values, the order of evaluation would be B, D, E and C. That is, the algorithm would not know that C is undefeated and would lose the opportunity for a quick pruning. This is understandable and does not undermine the usefulness of this approach, as shown by empirical testing.

Definition 8 (Dialectical bonsai) Let $\mathcal{T}(\mathsf{A})$ be an active dialectical tree in the context of a DAF τ, a **dialectical bonsai** $\mathcal{B}(\mathsf{A})$ for $\mathcal{T}(\mathsf{A})$ is a pruning of $\mathcal{T}(\mathsf{A})$ with a set of arguments $\mathsf{S_b}$ verifying:

$$\mathfrak{m}(\mathsf{B}_i, \lambda_k, \mathcal{T}(\mathsf{A})) = \mathfrak{m}(\mathsf{B}_i, \lambda_i, \mathcal{B}(\mathsf{A})), \text{ for every } \mathsf{B}_i \in \mathsf{S_b},$$

where λ_k is an argumentation line of $\mathcal{T}(\mathsf{A})$ containing B_i, λ_i is an argumentation line of $\mathcal{B}(\mathsf{A})$ containing B_i, and λ_k, λ_i are such that the paths from A to B_i coincide in both lines.

Proposition 2 *A 1U pruning for an active dialectical tree $\mathcal{T}$ is a dialectical bonsai for $\mathcal{T}$.*

Proof See Appendix. □

By definition, a dialectical bonsai is not an arbitrary pruning of its associated (active) dialectical tree, but one that shares the mark of every argument; in particular, the mark of the root. A more relaxed version of this definition could require sharing only the mark of the root argument. However, permitting a different marking for the rest of the arguments adds unnecessary complexity, and it constitutes ongoing research. The most important property dialectical bonsai should satisfy is to warrant exactly the same arguments than non-pruned trees do.

Lemma 1 (Soundness & completeness) *Given a dialectical bonsai $\mathcal{B}(\mathsf{A})$ of a dialectical tree $\mathcal{T}(\mathsf{A})$, A is warranted from $\mathcal{T}(\mathsf{A})$ iff A is warranted from $\mathcal{B}(\mathsf{A})$.*

Proof See Appendix. □

A different version of the definition for a dialectical bonsai might imply a much more complicated procedure to determine the status of the root argument. However, it would be desirable for any alternative definition to not interfere with the satisfaction of the meta-properties of soundness and completeness.

Note that, for the kind of pruning we are considering here, any heuristics would yield trees that are both sound and complete. Even a very poor heuristics would do so. This happens because the heuristics determines just the order of evaluation. On the other hand, the mechanism used for pruning could make the method to lose these two meta-properties. Or it could as well lead to even smaller bonsai. In any case this article is devoted to analyse pruning techniques that preserve the marking of all nodes, thus verifying Lemma 1. Next we introduce a particular notion of bonsai, whose pruning is based on argument strength and that happens to be a 1U pruning.

Fast-prune bonsai When a dialectical tree is built, the order in which children (i.e., attackers) are generated is relevant, as it could lead to very different results if a pruning technique like 1U were applied (as shown in Example 2). In our approach, each argument has an associated strength value, therefore once all the children of a given inner node were gathered, they are sorted from the strongest to the weakest. In this way, we always seek for the strongest attackers first in order to find an undefeated argument as fast as possible, to then be able to cut the remaining siblings off. This strategy is called *fast prune*.

Definition 9 (Fast-prune bonsai) Given a DAF τ and a dialectical bonsai $\mathcal{B}(\mathsf{A})$ for an active dialectical tree $\mathcal{T}(\mathsf{A})$, let B be an inner node in $\mathcal{B}(\mathsf{A})$ with a set of attackers Γ in $\mathcal{T}(\mathsf{A})$ such that the subset of attackers in $\mathcal{B}(\mathsf{A})$ is $\Gamma' \subseteq \Gamma$. Given the argument strength function $\mu(\cdot)$, $\mathcal{B}(\mathsf{A})$ is a **fast-prune bonsai** iff $\exists \mathsf{B}_k \in \Gamma', \mathfrak{m}(\mathsf{B}_k, \lambda_k, \mathcal{B}(\mathsf{A})) = \mathbf{U}$ implies both:

1. $\forall \mathsf{B}_{j \neq k} \in \Gamma', \mu(\mathsf{B}_k) \leq \mu(\mathsf{B}_j), \mathfrak{m}(\mathsf{B}_j, \lambda_j, \mathcal{B}(\mathsf{A})) = \mathbf{D}$
2. $\forall \mathsf{B}_x \in \Gamma \setminus \Gamma', \mu(\mathsf{B}_x) \leq \mu(\mathsf{B}_k)$.

Another way of reading this definition is to think that the strongest attackers B_j for B have to be considered within $\mathcal{B}(\mathsf{A})$, until an undefeated attacker is found; then, those attackers B_x that were left out of the bonsai are weaker than (or as strong as) the attackers for B in $\mathcal{B}(\mathsf{A})$. As in Definition 6, Γ' containing all defeated arguments receives no special treatment, since any subset would preserve the marking for B. Procedurally, however, the algorithm would have to check all the arguments, from the strongest to the weakest, to finally find out that none of them is undefeated. Hence, in practice, when all arguments in Γ are defeated, it holds that $\Gamma' = \Gamma$. Another variant would be to stop checking arguments statuses whenever a certain strength threshold is met, e.g., not to check arguments with a strength value below 0.2. In that case, we would have that $\Gamma \subset \Gamma$ and could prune even further, at the risk of losing soundness.

Proposition 3 *A fast-prune bonsai for a dialectical tree $\mathcal{T}$ is a 1U pruning for $\mathcal{T}$.*

Proof See Appendix. □

Example 5 Consider the potential tree for A depicted in Fig. 5a. Assuming that D and H are not active at the current time, we have the active tree shown in Fig. 5b. Applying fast prune over this active tree (which keeps the strength values from the potential) we have that the strongest attacker of A is C. This argument has two

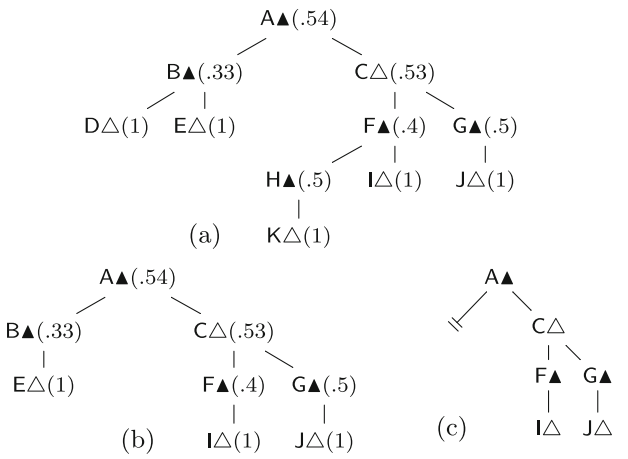


Fig. 5 **a** Potential tree for argument A in Example 5, **b** active tree for A for an active scenario and **c** fast-prune bonsai for A in the same active scenario

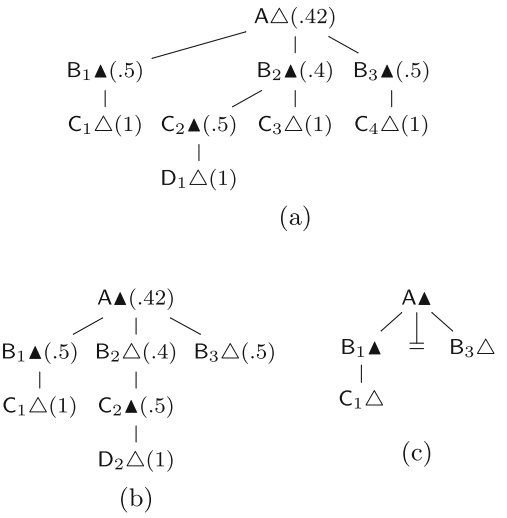


Fig. 6 **a** Potential tree for argument A in Example 6, **b** active tree for A for an active scenario and **c** fast-prune bonsai for A in the same active scenario

children, where G is the strongest one. Then, we consider G, whose only attacker is J, therefore G ends up defeated and its siblings should be evaluated. Again, F is defeated by leaf I, and C has no more attackers, thus it ends up undefeated. This means that C's siblings can be pruned off, yielding the bonsai in Fig. 5c. Note that every node in the bonsai preserves its marking; in particular the root.

Example 6 Consider the potential tree depicted in Fig. 6a. Assuming a world in which C_3 and C_4 are inactive, we have the active tree in Fig. 6b, where A is now defeated. The strongest attackers for A are B_1 and B_3. Following the lexicographical order, B_1 is expanded, whose one attacker C_1 is undefeated, hence B_1 is defeated. Then, we seek for the second stronger attacker for A, which is B_3, and it is a leaf, i.e., it is undefeated. Therefore, B_2 is cut off and the root is marked as defeated. The resulting bonsai is depicted in Fig. 6c. Note that the bonsai could get even smaller if it were composed just by A and B_3.

5 Empirical results

The previous section defined how to build a dialectical bonsai. An argumentation-based system using fast-prune bonsai for computing warrant would first precompile (only once) all strength values and then, in response to queries upon the changing set of active arguments, build and prune the corresponding active dialectical tree/s.

In this section, the performance of the fast-prune bonsai (FP) is measured from a simulation. The analysis involves comparing the FP bonsai against a *blind-prune bonsai* (BP), i.e., a pruning technique imitating the behaviour of a typical dialectical tree construction procedure, using no guide, like DeLP. The comparison between

the two bonsai will help us analyse the improvement achieved by FP bonsai. The simulation consists of the following steps:

1. *Generation of a random DAF with U arguments:* by creating a graph of U nodes and over U edges;
2. *Strength computation:* performed following the formula in Definition 7 using the algorithms in Section 3.2 by reutilising subtrees, which implies big time savings. Strength computation allows for storing trees (i.e., arguments and attacks), as a precompilation [10]. Since generally in a real-world scenario the construction of arguments takes time, we simulated this by introducing a small time penalty of 0.001 seconds each time an argument is built;
3. *Loop 500 times:*
   (a) *Deactivation of arguments:* a certain percentage of randomly chosen arguments is deactivaated.
   (b) *Selection of a query:* a random active argument is selected to act as a query.
   (b) *Computation of warrant:*
      – *Fast-prune bonsai:* an algorithm following Definition 9 computes warrant by using precompiled arguments and attacks; as for strength calculation, the time penalty per argument is introduced.
      – *Blind-prune bonsai:* an algorithm following the usual procedure in Definition 6 computes warrant by building arguments and attacks on-the-fly, which includes the time penalty.

Hence, each simulation generates a DAF, deactivates some arguments, and performs 500 queries. Since results are DAF-dependent, this simulation was ran 1000 times and the deactivation ratio was set to 10 %. The results for different DAF sizes can be seen in Table 1, where "U" indicates the size (i.e., number of arguments) of the DAF, "str(t)" the average amount of time taken to compute strength values, "FP(t)" (resp., "BP(t)") the average amount of seconds taken to compute the 500 queries using fast-prune (resp., blind-prune) bonsai, "B/F" indicates the *online speedup* obtained considering queries only (without time needed for *offline* strength computation), "str(#)" (resp., "BP(#)") reflects the average amount of arguments generated to compute strength (resp., BP bonsai).

A question that might arise after this description is why tolerating a time penalty while building the BP bonsai if the argument base can be precompiled. The answer to this is split in two: (1) we are trying to simulate the usual procedure to compute warrant (such as the one used in DeLP) which does not perform any precompilation; (2) precompilation of arguments and attacks involves computing potential trees and from there computation of strength would involve a slight overhead: traversing the tree from the leaves to the root. The original question however remains: how does FP perform against BP when both work over precompiled argument bases? The answer can be found in Table 2. Here, we consider the time taken to precompile knowledge,

Table 1 Results

U	str(t)	FP(t)	BP(t)	B/F	str(#)	BP(#)
200	0.86	0.029	2.02	70	807	1822
300	1.41	0.031	2.06	66	1329	1852
500	2.35	0.039	2.08	53	2219	1861

Table 2 Precompiled argument base

U	FP(t)	BP(t)	Bc/Fc
200	0.029	0.076	2.62
300	0.031	0.08	2.58
500	0.039	0.091	2.33

which is comparable to compute strength. Since both approaches would be storing potential trees, we took this element out of the equation. Speedup ("Bc/Fc") follows the same intuition as before. Note that the time BP bonsai ("BP(t)") take to respond to queries no longer has the tree-construction penalty, thus modifying the per-query speedup shown before("B/F").

As mentioned in Section 2, a scenario that would be favoured by the usage of FP bonsai is that of an argumentation-based agent. When looking at Table 1, speedup speaks by itself, whereas by considering precompiled KBs in Table 2 speedup is smaller but still largely justifies using FP bonsai to further maximise pruning, thus minimising time devoted to query answering.

Regarding the amount of space needed to store potential trees, we only need to store the potential graph, as each potential tree is the spanning of this graph from a given argument. For instance, a DAF of 500 arguments and 500 attacks with a maximum argument size of N kilobytes would require $500 \times N + 500 \times M$, where M is just a few bytes representing an attack: both arguments IDs plus attack direction. For this problem size, considering arguments of a maximum size of 5 kb (which is approximately two pages of plain text), the amount of storage necessary to keep the precompiled information would remain below 5 mb.

6 Conclusions

We have presented an approach to accelerate the computation of dialectical trees (and thus warrant) within a dynamic abstract argumentative setting. Such dynamic frameworks allow for the representation of active and inactive arguments, being the former the only ones to be considered to compute warrant. This characteristic leads to the consideration of potential dialectical trees (containing every argument in the universal set), as well as active trees (containing only currently active arguments). Therefore, every time the situation changes, warrants would have to be recalculated. There is a high degree of uncertainty regarding how the fluctuation of active arguments affects previously computed warrants. Nonetheless, the information codified by potential trees can be successfully used to speed up the construction of active trees. To this end, we calculate a measure of strength for each argument using potential trees. This is an advantage, as strengths are calculated just once, and not upon every change in the world, as it would be done if strength were calculated over active trees. Strength computation, even when performed once, is indeed expensive. However, empirical testing has shown that the time taken to compute argument strength is amortised within a reasonable time window. Another advantage of pruned trees is that of readability; smaller trees are easier to visualise. This is another motivation for finding the smallest possible trees, as they justify the mark of the root by using the minimal amount of arguments.

We presented a comprehensive set of postulates describing the desirable behaviour of any strength formula. The concrete measure presented in Section 3.2 was successfully checked against these postulates. Having such a characterisation formalises the expected behaviour of any given strength measure; this eases the definition of new strength formulas.

The fast-prune heuristic technique is guided by argument strength and attempts to maximise pruning by looking to find undefeated arguments as soon as possible, which implies that their siblings can be cut off. Computing argument strength allows us to store precompiled arguments and attacks, which would be used later when answering queries. Hence, significant speedup (shown as "B/F" in Table 1) is not only a result of traversing smaller trees, but also due to the non-calculation of arguments and attacks. In the case of fast prune (FP), arguments are built at the stage of strength computation; as for blind prune (BP), every query triggers the construction of a tree and every argument in it. Table 1 shows both quantities in columns "str(#)" and "BP(#)", where the latter is the accumulated amount after 500 queries. Finally, when considering precompiled arguments and attacks for both approaches, speedup remains considerable, as illustrated by column "Bc/Fc" in Table 2, where is shown that FP responds generally twice as fast than BP.

Experimental results indicate that each fast-prune bonsai is a significantly smaller version of its associated non-pruned active tree, providing a meaningful speedup. This plus precompilation of arguments and attacks is an asset in massive argumentation domains, like the WWW [21], where repeatedly looking for counter-arguments is expensive. Another scenario where our approach would perform well can be found in the context of multi-agent systems. An argumentation-based agent with goals expressed as a set of warranted arguments [1] would be computing warrant several times per cycle. In such a setting, precompiling and storing potential trees would yield a twice-as-fast response.

7 Related work

The problem of efficiently computing the marking of the argument in the root of a dialectical tree has received scarce attention. In [13] the problem was already addressed; however, the approach is mostly declarative and does not include any concrete techniques to perform the pruning, leaving out the definition of interesting heuristics or empirical testing. Meanwhile, the approach taken in [15] was a concrete attempt to address the issues related to the implementation of processes related to the computation of argument status. In general, there has been little attention devoted to this important problem.

In [8] an approach for speeding up the construction of argumentation trees is presented. The authors also use information from the argument knowledge base beforehand, to speed up the argumentation process. This argument compilation produces a hyper-graph of the inconsistent subsets, which, together with a special algorithm, can be used for efficiently building argument trees, saving time for conflict detection. In our work, we try to avoid the evaluation of attackers whenever possible. Both approaches follow parallel paths and could be combined.

In order to deal with warrant recalculation, some work has been recently done on dynamics in argumentation. In [9], the authors propose a series of principles to

determine under what conditions an extension does not change when faced to a change in the framework. A similar article [12] analyses the impact on the set of extensions provoked by the addition of a single argument to a framework. Another paper [5] proposes possibility and impossibility results regarding argument additions and enforcement of a set of warranted arguments. These articles are complementary to ours, as they define properties that operate on top of an argumentation framework that is assumed to change. Our method to accelerate warrant computation could be used in tandem with these methods.

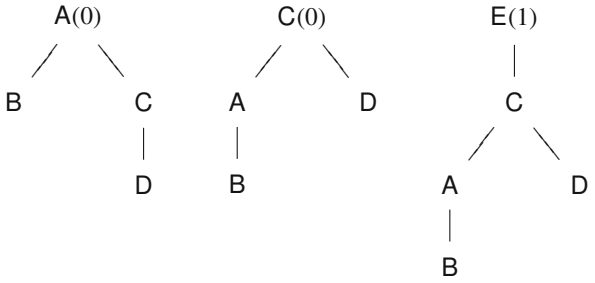
Concerning the relation between Dung semantics and the warrant given by our adopted marking criterion, there is a resemblance with the grounded extension. The difference is that our approach does not allow arguments to be repeated within an argumentation line, whereas the grounded semantics only prevents pro arguments from being repeated. For instance, assume an AF $(\{A, B, C\}, \{B \rightarrow A, B \rightarrow C, C \rightarrow B\})$. In the DAF resulting from considering all arguments as active, the dialectical tree would be the line $A - B - C$, whereas for Dung the tree spanning from the graph would be $A - B - C - B$. In the former A would be accepted, while it is rejected in the latter.

The work presented in [18] is related to ours by the fact that introduces a computationally oriented method to update the semantics of an abstract argumentation framework when the set of arguments or the attack relation change; the division-based procedure is designed to improve the computational effectiveness of such an update. This method effects a separation of the framework in three sets of arguments (unaffected, affected, and the connecting parts) providing a way of handling the change in the framework in such a way that the computation of the new semantics can be done in polynomial time; and, when this cannot be achieved, an exponential improvement in the complexity can be obtained. Clearly, the approach presented in [18] differs from ours. On the one hand, our approach is based on dialectical trees while theirs is based on the extensional approach for argumentative semantics over graphs of arguments. On the other hand, our pruning technique is guided by precalculated argument strength values that are afterwards used for any configuration of active arguments. In contrast, they propose an efficient method to compute the arguments statuses of an updated argumentation framework by just considering the part of the framework affected by this update. Since both approaches involve dynamics and pursue efficiency, it could be interesting to study the feasibility of combining them. In particular, we could study how to use a heuristic approach on their framework. Symmetrically, we could study how to apply the separation process to our proposal by analysing how an active instance transforms into another.

8 Future work

Apart from the argument strength formula presented in Section 3.2, other formulas could be considered. For instance, a formula stating that an argument A is as strong as its support, which is understood as the difference between pro and con arguments in the potential tree for A. An argument with no defeaters would be assumed to have infinite support and assigned ∞ by default. Note that there would not be local strength values, as the actual strength calculation is a simple count. For example,

applying this formula to the DAF in Example 3 would yield the following strength values:



Analysing this and other strength formulas has been left as future work, as presenting them formally and checking compliance with the postulates presented in Section 3.1 would have diverted the scope of the present article.

Our approach does not reach the "optimal" bonsai. That is, although the heuristics clearly accelerates computation of warrant by analysing a smaller number of arguments, it cannot ensure bonsai minimality. Addressing this challenge is a matter for future work. Another optimisation to be analysed is to find a new heuristics in which local strength values and actual strength values always coincide.

We plan to apply the pruning technique to the variation of the DeLP system used in [10]. This system contains defeasible rules only and specifies a set of "perceptions" (i.e., facts) that changes dynamically. One of the main issues is to find a way to (efficiently) build all potential arguments. Although this should consider any possible set of facts that could arise, operationally defeasible rules are analysed one by one. Each possible chaining of rules gives place to a new potential argument. Potential trees would be used to compute strength. Once this has been done, given a particular set of perceptions, a subset of these potential arguments will be active and thus the method studied in this article could be used to prune DeLP dialectical trees. More difficulties have to be addressed for this rule-based approach to work; for instance, potential arguments would be uninstantiated. That is, they are schemes and each one of them could lead to several active arguments.

For certain argumentation frameworks the initial computation of strength could be too expensive. In such a case, instead of computing strength values using potential trees, an alternative approach can be implemented. The first query for a given argument A could make the system to build a complete (i.e., non-pruned) dialectical tree to find the answer. This tree would be also used to compute the strength value for A. Both the tree and the strength value are stored. Subsequent queries for arguments other than A but involving A in their active tree would use this strength value, as if it were computed from a potential tree. Subsequent queries for A could refine the strength value by completing the pseudo-potential tree. Over time, the performance of this method would be increasingly better than pruning blindly.

The application of our approach to generalised Dung semantics indeed represents future work. In order to achieve this, firstly we would need to de-attach the strength formula from dialectical trees to make it compatible with argumentation graphs. As mentioned in the previous section, our semantics is very close to Dung's grounded semantics; therefore, a straightforward extension could be achieved by relaxing the occurrence check over con arguments within an argumentation line. In this case, the

rest of the theoretical elements (strength formula, pruning policy, etc.) we propose would remain valid.

Appendix

Proposition 1 *The strength measure $\mu(\cdot)$ from Definition 7 satisfies the postulates* (***S1***)*,* (***S2***)*,* (***S3***)*,* (***S4***)*,* (***S5***) *and* (***S6***)*.*

Proof Let $\mu(\cdot)$ be the strength measure of Definition 7, τ a DAF, $\mathcal{T}_1$ and $\mathcal{T}_2$ two potential trees from τ, and two arguments A_1 and A_2 placed in an arbitrary position in $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Also let $\mathcal{T}_1^{\Delta}$ be a potential tree that is the result of arbitrarily adding and/or subtracting arguments to the subtree of $\mathcal{T}_1$ rooted in A_1.

(**S1**) $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2)$ iff $\mu_{\mathsf{atts}}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \leq \mu_{\mathsf{atts}}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$.
Let $\mathsf{B}_1, \ldots, \mathsf{B}_n$ be the attackers of A_1 in λ_1 of $\mathcal{T}_1$ and $\mathsf{C}_1, \ldots, \mathsf{C}_n$ be the attackers of A_2 in λ_2 of $\mathcal{T}_2$.
If $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2)$, then by Definition 7:

$$\frac{1}{\sum \mu(\mathsf{B}_i, \lambda_i, \mathcal{T}_1) + 1} \leq \frac{1}{\sum \mu(\mathsf{C}_j, \lambda_j, \mathcal{T}_2) + 1}$$

$$\sum \mu(\mathsf{B}_i, \lambda_i, \mathcal{T}_1) + 1 \geq \sum \mu(\mathsf{C}_j, \lambda_j, \mathcal{T}_2) + 1$$

$$\sum \mu(\mathsf{B}_i, \lambda_i, \mathcal{T}_1) \geq \sum \mu(\mathsf{C}_j, \lambda_j, \mathcal{T}_2)$$

Then it holds that $\mu_{\mathsf{atts}}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \leq \mu_{\mathsf{atts}}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$.

(**S2**) if $\mathsf{atts}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) = \emptyset$ and $\mathsf{atts}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \neq \emptyset$ then $\mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) < \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$. Let $\mathsf{C}_1, \ldots, \mathsf{C}_n$ be the attackers of A_2 in λ_2 of $\mathcal{T}_2$. By hypothesis $\mathsf{atts}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) = \emptyset$ then by Definition 7 $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) = 1$, and since $\mathsf{atts}(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) \neq \emptyset$ then by Definition 7 $\sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_2) > 0$. Thus, $1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_2) > 1$, and it holds that

$$\frac{1}{1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_2)} < 1$$

Then by Definition 7 it holds that $\mu(\mathsf{A}_2, \lambda_2, \mathcal{T}_2) < \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)$.

(**S3**) if $|\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \leq |\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ and $|\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \geq |\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ then $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$.
By induction over n = "the amount of changes performed over $\mathcal{T}_1$ to obtain $\mathcal{T}_1^{\Delta}$, where a change is an addition or a removal of a subtree".

- **Initial Step:** When $n = 0$ that is $\mathcal{T}_1 = \mathcal{T}_1^{\Delta}$ therefore, $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) = \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$. Then, $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ holds.
- **Inductive Step:** Let us assume that $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1{}^{\Delta-1})$ holds with $\mathcal{T}_1{}^{\Delta-1}$ having $n-1$ changes with respect to $\mathcal{T}_1$, and let us prove that $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ holds for $\mathcal{T}_1^{\Delta}$ having one change with respect to $\mathcal{T}_1{}^{\Delta-1}$. Then, by hypothesis we have two cases: either this change is an

addition of a pro argument or the removal of a con argument (at any case, the change involves the corresponding subtree):

- If $\mathcal{T}_1^{\Delta}$ adds a pro argument E, then there is an argument C such that $\mathsf{C} \in \textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta-1})$ and $\mathsf{C} \in \textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, assuming that $\mathsf{D}_1 \ldots, \mathsf{D}_k$ are C's attackers on $\mathcal{T}_1^{\Delta-1}$ and $\mathsf{D}_1 \ldots, \mathsf{D}_k, \mathsf{E}$ are C's attackers on $\mathcal{T}_1^{\Delta}$. By inductive hypothesis $\sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta-1}) = \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta})$, and by Definition 7 $\mu(\mathsf{E}, \lambda_j, \mathcal{T}_1^{\Delta}) \geq 0$. Therefore,

$$\frac{1}{1 + \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta-1})} \geq \frac{1}{1 + \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta}) + \mu(\mathsf{E}, \lambda_j, \mathcal{T}_1^{\Delta})}$$

  which implies by Definition 7 that $\mu(\mathsf{C}, \lambda_m, \mathcal{T}_1^{\Delta-1}) \geq \mu(\mathsf{C}, \lambda_m, \mathcal{T}_1^{\Delta})$. Since C's strength is reduced in $\mathcal{T}_1^{\Delta}$ with respect to $\mathcal{T}_1^{\Delta-1}$ and the only change in $\mathcal{T}_1^{\Delta}$ is the addition of E, the only strengths that will be affected are those of C's ancestors. In particular, by Definition 7 the strength of C's father F will be increased in $\mathcal{T}_1^{\Delta}$, which also decrements the strength of F's father G. Given that $\mathsf{C} \in \textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, then $\mathsf{F} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ and $\mathsf{G} \in \textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$. Following, the strength of C's ancestors which are A's Pro arguments will increase in $\mathcal{T}_1^{\Delta}$ with respect to $\mathcal{T}_1^{\Delta-1}$. Since $\mathsf{A} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, it holds that $\mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta-1}) \leq \mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta})$. Thus, by inductive hypothesis $\mu(\mathsf{A}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta})$ holds.
- If $\mathcal{T}_1^{\Delta}$ removes a con argument E, then there is an argument C such that $\mathsf{C} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta-1})$ and $\mathsf{C} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, assuming that $\mathsf{D}_1 \ldots, \mathsf{D}_k, \mathsf{E}$ are C's attackers on $\mathcal{T}_1^{\Delta-1}$ and $\mathsf{D}_1 \ldots, \mathsf{D}_k$ are C's attackers on $\mathcal{T}_1^{\Delta}$. By inductive hypothesis $\sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta-1}) = \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta})$, and by Definition 7 $\mu(\mathsf{E}, \lambda_j, \mathcal{T}_1^{\Delta}) \geq 0$. Therefore,

$$\frac{1}{1 + \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta-1}) + \mu(\mathsf{E}, \lambda_j, \mathcal{T}_1^{\Delta})} \leq \frac{1}{1 + \sum \mu(\mathsf{D}_i, \lambda_i, \mathcal{T}_1^{\Delta})}$$

  which implies by Definition 7 that $\mu(\mathsf{C}, \lambda_m, \mathcal{T}_1^{\Delta-1}) \leq \mu(\mathsf{C}, \lambda_m, \mathcal{T}_1^{\Delta})$. Since C's strength is augmented in $\mathcal{T}_1^{\Delta}$ with respect to $\mathcal{T}_1^{\Delta-1}$ and the only change in $\mathcal{T}_1^{\Delta}$ is the removal of E, the only strengths that will be affected are those of C's ancestors. In particular, by Definition 7 the strength of C's father F will be decreased in $\mathcal{T}_1^{\Delta}$, which also increases the strength of F's father G. Given that $\mathsf{C} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, then $\mathsf{F} \in \textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ and $\mathsf{G} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$. Following, the strength of C's ancestors which are A's Pro arguments will increase in $\mathcal{T}_1^{\Delta}$ with respect to $\mathcal{T}_1^{\Delta-1}$. Since $\mathsf{A} \in \textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$, it holds that $\mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta-1}) \leq \mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta})$. Thus, by inductive hypothesis $\mu(\mathsf{A}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}, \lambda_1, \mathcal{T}_1^{\Delta})$ holds.

(S4) if $|\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \leq |\textsc{con}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ and $|\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1)| \geq |\textsc{pro}(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})|$ then $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$.
Analogous to (**S3**).

(**S5**) if $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ then

(a) $(\forall \mathsf{B} : \mathsf{A}_1 \in \textsc{pro}(\mathsf{B}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1^{\Delta})$ and

(b) $(\forall \mathsf{C} : \mathsf{A}_1 \in \textsc{con}(\mathsf{C}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1^{\Delta})$.

The proof for (a) is by induction over n = "the number of pro ancestors of A_1 in the argumentation line λ_1 of $\mathcal{T}_1$ and $\mathcal{T}_1^{\Delta}$".

- **Initial Step:** When $n = 0$ the argument A_1 is the only argument that has A_1 as a pro argument in the line λ_1 of $\mathcal{T}_1$ and $\mathcal{T}_1^{\Delta}$. Then it holds by hypothesis that $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$.
- **Inductive Step:** Let us assume that $\mu(\mathsf{B}_{n-1}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{B}_{n-1}, \lambda_1, \mathcal{T}_1^{\Delta})$ holds with B_{n-1} the (n-1)th pro ancestor of A_1 in λ_1, and let us prove that this holds for B_n the nth pro ancestor of A_1 in λ_1. Let $\mathsf{C}_1 \ldots \mathsf{C}_k, \mathsf{D}$ be the attackers of B_n in $\mathcal{T}_1^{\Delta}$ and $\mathcal{T}_1$ where D is the attacker that appears in λ_1, and let $\mathsf{E}_1, \ldots, \mathsf{E}_n, \mathsf{B}_{n-1}$ be the set of attackers of D in $\mathcal{T}_1^{\Delta}$ and $\mathcal{T}_1$. By definition of $\mathcal{T}_1^{\Delta}$ $\sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1) = \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1^{\Delta})$ and $\sum \mu(\mathsf{E}_i, \lambda_i, \mathcal{T}_1) = \sum \mu(\mathsf{E}_i, \lambda_i, \mathcal{T}_1^{\Delta})$. Then:

$$1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1) = 1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1^{\Delta})$$

$$1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1) + \mu(\mathsf{B}_{n-1}, \lambda_1, \mathcal{T}_1) \leq 1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1^{\Delta}) + \mu(\mathsf{B}_{n-1}, \lambda_1, \mathcal{T}_1^{\Delta})$$

[By Inductive Hypothesis]

$$\frac{1}{1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1^{\Delta}) + \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})} \geq \frac{1}{1 + \sum \mu(\mathsf{C}_i, \lambda_i, \mathcal{T}_1^{\Delta}) + \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})}$$

$$\mu(\mathsf{D}, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{D}, \lambda_1, \mathcal{T}_1^{\Delta})$$

[By Definition 7]

$$1 + \sum \mu(\mathsf{D}_j, \lambda_j, \mathcal{T}_1) + \mu(\mathsf{D}_1, \lambda_1, \mathcal{T}_1) \geq 1 + \sum \mu(\mathsf{D}_j, \lambda_j, \mathcal{T}_1^{\Delta}) + \mu(\mathsf{D}_1, \lambda_1, \mathcal{T}_1)$$

$$\frac{1}{1 + \sum \mu(\mathsf{D}_j, \lambda_j, \mathcal{T}_1) + \mu(\mathsf{D}_1, \lambda_1, \mathcal{T}_1)} \leq \frac{1}{1 + \sum \mu(\mathsf{D}_j, \lambda_j, \mathcal{T}_1^{\Delta})}$$

$$\mu(\mathsf{B}_n, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{B}_n, \lambda_1, \mathcal{T}_1^{\Delta})$$

[By Definition 7]

The proof for (b) is analogous to (a) where n is "the number of con ancestors of A_1 in λ_1" and the initial step considers the father of A_1.

(S6) if $\mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{A}_1, \lambda_1, \mathcal{T}_1^{\Delta})$ then
$(\forall \mathsf{B} : \mathsf{A}_1 \in \text{PRO}(\mathsf{B}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1) \geq \mu(\mathsf{B}, \lambda_1, \mathcal{T}_1^{\Delta})$ and
$(\forall \mathsf{C} : \mathsf{A}_1 \in \text{PRO}(\mathsf{C}, \lambda_1, \mathcal{T}_1))\ \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1) \leq \mu(\mathsf{C}, \lambda_1, \mathcal{T}_1^{\Delta})$.
Analogous to **(S5)**.

Therefore, the strength formula from Definition 7 verifies postulates **(S1)** to **(S6)** □

Proposition 2 *A 1U pruning for an active dialectical tree $\mathcal{T}$ is a dialectical bonsai for $\mathcal{T}$.*

Proof Let $\mathcal{P}(\mathsf{A})$ be a 1U pruning of a dialectical tree $\mathcal{T}(\mathsf{A})$. Since for any set of attackers for a given argument B in the original tree $\mathcal{T}(\mathsf{A})$ with at least one undefeated attacker, exactly one of these remains in the 1U pruning $\mathcal{P}(\mathsf{A})$ by Definition 6, then B is defeated both in $\mathcal{T}(\mathsf{A})$ and $\mathcal{P}(\mathsf{A})$. On the other hand, for any set of attackers for B where no argument is undefeated, any subset of attackers (even the empty set) would yield B as undefeated. □

Lemma 1 *Given a dialectical bonsai $\mathcal{B}(\mathsf{A})$ of a dialectical tree $\mathcal{T}(\mathsf{A})$, A is warranted from $\mathcal{T}(\mathsf{A})$ iff A is warranted from $\mathcal{B}(\mathsf{A})$.*

Proof Trivial, from Definition 8, the marking of the root nodes in $\mathcal{B}(\mathsf{A})$ equals the marking of the root in $\mathcal{T}(\mathsf{A})$:

($\Rightarrow$) By hypothesis, A is marked as **U** in $\mathcal{T}(\mathsf{A})$. By Definition 8, $\mathfrak{m}(\mathsf{A}, \lambda_k, \mathcal{T}(\mathsf{A})) = \mathfrak{m}(\mathsf{A}, \lambda_i, \mathcal{B}(\mathsf{A}))$, then $\mathfrak{m}(\mathsf{A}, \lambda_i, \mathcal{B}(\mathsf{A})) = \mathbf{U}$ and A is warranted in $\mathcal{B}(\mathsf{A})$.

($\Leftarrow$) By hypothesis, A is marked as **U** in $\mathcal{B}(\mathsf{A})$. By Definition 8, $\mathfrak{m}(\mathsf{A}, \lambda_k, \mathcal{T}(\mathsf{A})) = \mathfrak{m}(\mathsf{A}, \lambda_i, \mathcal{B}(\mathsf{A}))$, then $\mathfrak{m}(\mathsf{A}, \lambda_i, \mathcal{T}(\mathsf{A})) = \mathbf{U}$ and A is warranted in $\mathcal{T}(\mathsf{A})$.

Therefore, A is warranted from $\mathcal{T}(\mathsf{A})$ iff A is warranted from $\mathcal{B}(\mathsf{A})$. □

Proposition 3 *A fast-prune bonsai for a dialectical tree $\mathcal{T}$ is a 1U pruning for $\mathcal{T}$.*

Proof A fast-prune bonsai requires, for any set of attackers with at least one undefeated argument: (1) to have only one undefeated argument B_k (2) for B_k to be weaker than its (defeated) siblings in the bonsai. Condition 1 is equivalent to the requirement for a 1U pruning. □

References

1. Amgoud, L., Devred, C., Lagasquie-Schiex, M.C.: A constrained argumentation system for practical reasoning. In: AAMAS, pp. 429–436 (2008)
2. Amgoud, L., Dimopoulos, Y., Moraitis, P.: A general framework for argumentation-based negotiation. In: ArgMAS, pp. 1–17 (2007)
3. Baroni, P., Dunne, P.E., Giacomin, M.: Computational properties of resolution-based grounded semantics. In: IJCAI, pp. 683–689 (2009)
4. Baroni, P., Giacomin, M.: On principle-based evaluation of extension-based argumentation semantics. Artif. Intell. **171**, 675–700 (2007)
5. Baumann, R., Brewka, G.: Expanding argumentation frameworks: Enforcing and monotonicity results. In: COMMA, pp. 75–86 (2010)
6. Bench-Capon, T.J.M., Dunne, P.E.: Argumentation in artificial intelligence. Artif. Intell. **171**, 619–641 (2007)

7. Besnard, P., Hunter, A.: A logic-based theory of deductive arguments. Artif. Intell. **128**(1–2), 203–235 (2001)
8. Besnard, P., Hunter, A.: Knowledgebase compilation for efficient logical argumentation. In: KR, pp. 123–133 (2006)
9. Boella, G., Kaci, S., van der Torre, L.: Dynamics in argumentation with single extensions: abstraction principles and the grounded extension. In: ECSQARU, pp. 107–118 (2009)
10. Capobianco, M., Chesñevar, C.I., Simari, G.R.: Argumentation and the dynamics of warranted beliefs in changing environments. JAAMAS **11**, 127–151 (2005)
11. Cayrol, C., Lagasquie-Schiex, M.C.: Graduality in argumentation. J. Artif. Intell. Res. (JAIR) **23**, 245–297 (2005)
12. Cayrol, C., de Saint-Cyr, F.D., Lagasquie-Schiex, M.C.: Change in abstract argumentation frameworks: adding an argument. J. Artif. Intell. Res. (JAIR) **38**, 49–84 (2010)
13. Chesñevar, C.I., Simari, G.R., Godo, L.: Computing dialectical trees efficiently in possibilistic defeasible logic programming. In: LPNMR, pp. 158–171 (2005)
14. Chesñevar, C., Simari, G.: A lattice-based approach to computing warranted belief in skeptical argumentation frameworks. In: IJCAI, pp. 280–285 (2007)
15. Chesñevar, C.I., Simari, G.R., García, A.J.: Pruning search space in defeasible argumentation. In: ATAI, pp. 46–55 (2000)
16. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artif. Intell. **77**(2), 321–358 (1995)
17. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. TPLP **4**(1–2), 95–138 (2004)
18. Liao, B., Jin, L., Koons, R.C.: Dynamics of argumentation systems: a division-based method. Artif. Intell. **175**(11), 1790–1814 (2011)
19. Matt, P.A., Toni, F.: A game-theoretic measure of argument strength for abstract argumentation. In: JELIA, pp. 285–297 (2008)
20. Prakken, H., Vreeswijk, G.: Logics for defeasible argumentation. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, 2nd edn, vol. 4, pp. 219–318. Dordrecht etc. (2002)
21. Rahwan, I.: Mass argumentation and the semantic web. J. Web. Sem. **6**(1), 29–37 (2008)
22. Rahwan, I., Zablith, F., Reed, C.: Towards large scale argumentation support on the semantic web. In: AAAI, pp. 1446–1451 (2007)
23. Reed, C., Wells, S., Devereux, J., Rowe, G.: Aif+: dialogue in the argument interchange format. In: COMMA, pp. 311–323 (2008)
24. Rotstein, N., Moguillansky, M., García, A., Simari, G.: A dynamic argumentation framework. In: COMMA, pp. 427–438 (2010)
25. Rotstein, N.D., Gottifredi, S., García, A.J., Simari, G.R.: A heuristics-based pruning technique for argumentation trees. In: SUM, pp. 177–190 (2011)
26. Rotstein, N.D., Moguillansky, M.O., Simari, G.R.: Dialectical abstract argumentation: a characterization of the marking criterion. In: IJCAI, pp. 898–903 (2009)
27. Wooldridge, M.J.: An Introduction to MultiAgent Systems, 2nd edn. John Wiley & Sons (2009)