Computing Argument Preferences and Explanations in Abstract Argumentation

Quratul-ain Mahesar
Department of Computing Science, University of Aberdeen, Aberdeen, UK
Email: quratul-ain.mahesar@abdn.ac.uk

***Abstract*—We present a novel automated approach for the computation and verification of preferences in an abstract argumentation system. Various argumentation semantics have been developed for identifying acceptable sets of arguments, however, there is a lack of explanatory justification for their acceptability based on preferences. We present an algorithm which takes an abstract argumentation framework and a single extension (conflict-free set of arguments) as input, and outputs preference relations that explain why a set of arguments are acceptable as opposed to their attackers. We also present an algorithm to verify that the output preferences when used with the given argumentation framework induce the input extension.**

Keywords: argumentation, preferences, reasoning, explanation.

I. Introduction

Preferences play a central part in decision making and have been extensively studied in various disciplines such as economy, operations research, psychology and philosophy [16]. Preferences are used in many areas of artificial intelligence including planning, scheduling, multi-agent systems, combinatorial auctions and game playing [21]. The complexity of eliciting preferences and representational questions like dealing with uncertainty has remained a very active research area [13], [17], [21]. Logic based abstract argumentation [9] provides a formal representation of preferences. Argumentation has gained an increasing popularity in Artificial Intelligence (AI). It has been widely used for handling inconsistent knowledge bases [7], [11], [19], and dealing with uncertainty in decision making [3], [8], [15].

An abstract argumentation framework [9] is a directed graph consisting of nodes that represent unique atomic arguments and directed edges that represent an attack between two arguments. This visual representation of an argumentation framework as a directed graph is also known as an argumentation graph. Acceptable sets of arguments called extensions for an argumentation framework can be computed based on various acceptability semantics [9]. Arguments can have different strengths, e.g., an argument relies on more certain or important information than another. This has led to the introduction of preference-based argumentation framework consisting of preference relations between arguments [1]. Furthermore as given in [4], preferences are taken into account in the evaluation of arguments at the semantic level, which is also known as preference-based acceptability [2]. The basic idea is to accept undefeated arguments and also arguments that are preferred to their attacking arguments, as these arguments can defend themselves against their attacking arguments.

Explainability is one of the key issues in AI systems that must interact naturally to support users in decision making. Systems need to be capable of explaining their output. Addressing this challenge is critical if we are to use AI with the intent of improving user performance and experience. Argumentation has been previously used for transparently explaining the procedure and the results of reasoning, for instance, [10] identify related information and generate an explanation for a topic through some fictitious debate game between two players. However, it only considers explanation of arguments that are related to each other. [22] present an algorithm to generate natural language explanations from debate trees, but the algorithm is domain specific and solely concerns admissibility.

In our research, we deal with both the issues of preference computation; and explanation of the reasoning process. We provide an automated approach to compute argument preferences from an abstract argumentation framework and an extension (consisting of conflict-free arguments), that explain why an argument is in an extension as opposed to its attacking argument(s). We present a novel algorithmic-approach to identify the arguments that survive from attacks, thereby computing preferences for such arguments. We categorize the arguments based on whether they survive by direct defence, i.e., an argument appears in an extension despite being attacked by another argument by defending itself; or indirect defence, an argument is defended by another argument with both these arguments being present in the extension. Furthermore, we present an algorithm for the verification of the computed preferences.

The rest of this paper is organised as follows. In Section II, we present the background on abstract argumentation framework and acceptability semantics for acceptable set of arguments also known as extensions. In Section III, we present the background on preference-based argumentation framework, and we present two types of preferences and their representation. In Section IV, we present our proposed algorithms for computing preferences and verifying them. Implementation details along with the complete flow chart of the automated approach are given in Section V. Finally, we conclude in Section VI.

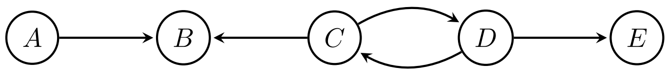

Figure 1. Example abstract argumentation framework AAF_1

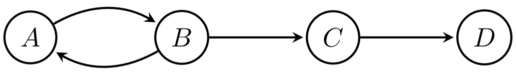

Figure 2. Example abstract argumentation framework AAF_2

II. Preliminaries

In this section we briefly summarise the background information related to classical Abstract Argumentation Frameworks (AAFs).

Definition II.1. *(Abstract Argumentation Framework [9]): An abstract argumentation framework (AAF) is a pair $AAF = (\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}$ is an attack relation ($\mathcal{R} \subseteq \mathcal{A} \times \mathcal{A}$). The notation $(A, B) \in \mathcal{R}$ where $A, B \in \mathcal{A}$ denotes that A attacks B.*

To calculate the sets of arguments that can be accepted together - called extensions - different acceptability semantics are introduced in the literature, and from these the most common are given as follows [9].

Definition II.2. *(Extensions): Let $AAF = (\mathcal{A}, \mathcal{R})$ be an abstract argumentation framework, $\mathcal{E} \subseteq \mathcal{A}$ and $A, B, C \in \mathcal{A}$*

- *$\mathcal{E}$ is conflict free iff there exists no arguments $A, B \in \mathcal{E}$ such that $(A, B) \in \mathcal{R}$.*
- *$\mathcal{E}$ is admissible iff it is conflict free and defends all its arguments. $\mathcal{E}$ defends A iff for every argument $B \in \mathcal{A}$, if we have $(B, A) \in \mathcal{R}$ then there exists $C \in \mathcal{E}$ such that $(C, B) \in \mathcal{R}$.*
- *$\mathcal{E}$ is a complete extension iff $\mathcal{E}$ is an admissible set which contains all the arguments it defends.*
- *$\mathcal{E}$ is a preferred extension iff it is a maximal (with respect to set inclusion) admissible set.*
- *$\mathcal{E}$ is a stable extension iff it is conflict-free and for all $A \in \mathcal{A} \setminus \mathcal{E}$, there exists an argument $B \in \mathcal{E}$ such that $(B, A) \in \mathcal{R}$.*
- *$\mathcal{E}$ is a grounded extension iff $\mathcal{E}$ is a minimal (for set inclusion) complete extension.*

Example II.1. *Given the abstract argumentation framework of Figure 1, then we compute its extensions as follows:*

- *Conflict free: $\{A, C, E\}, \{A, D\}, \{B, D\}, \{A, C\}, \{A, E\}, \{B, E\}, \{C, E\}, \{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \emptyset$*
- *Admissible: $\{A, C, E\}, \{A, C\}, \{A, D\}, \{C, E\}, \{A\}, \{C\}, \{D\}, \emptyset$*
- *Complete: $\{A, C, E\}, \{A, D\}, \{A\}$*
- *Preferred: $\{A, C, E\}, \{A, D\}$*
- *Stable: $\{A, C, E\}, \{A, D\}$*
- *Grounded: $\{A\}$*

III. Preference Types and Representations

While an abstract argumentation framework captures the basic interactions between arguments, it does not consider factors such as argument strength, i.e., arguments may not necessarily have the same strengths [6], [19]. Consequently, preferences over arguments can be added to the argumentation framework and taken into account in order to evaluate arguments [1], [14], [18]. Preference-based argumentation framework [1] extends abstract argumentation framework to account for preferences over arguments. The attack relation in a preference-based argumentation framework is called defeat, and is denoted by Def.

Definition III.1. *(Preference-based Argumentation Framework (PAF) [1]):*
A preference-based argumentation framework is a triple $(\mathcal{A}, Def, \geq)$ where $\mathcal{A}$ is a set of arguments, Def is the defeat binary relation on $\mathcal{A}$, and $\geq$ is a (partial or total) pre-ordering defined on $\mathcal{A} \times \mathcal{A}$. The notation $(A, B) \in Def$ means that argument A defeats argument B.

The notation $A \geq B$ means that argument A is at least as preferred as B and the relation $>$ is the strict counterpart of $\geq$.

Example III.1. *Let there be the argumentation framework of Figure 2. Preferences could be applied in two ways [5]: one way is to apply preferences at the time of argument acceptability (semantic level); and second way is to compute all preferred extensions and filter them by the application of the preferences. By using the first method, if we assume $\{A > B, C > D\}$ is the set of preferences between arguments, then we get a single extension $\mathcal{E} = \{A, C\}$. Now, by using the second method, we first compute all preferred extensions $[\{A, C\}, \{B, D\}]$. These extensions could now be filtered by the application of the set of preferences $\{A > B, C > D\}$ which suggest $\{A, C\}$ to be better than $\{B, D\}$.*

A preference-based argumentation framework can represent an abstract argumentation framework [12]:

Definition III.2. *(PAF representing an AAF) A preference-based argumentation framework $(\mathcal{A}, Def, \geq)$ represents an abstract argumentation framework $(\mathcal{A}, \mathcal{R})$ iff $\forall A, B \in \mathcal{A}$, it is the case that $(A, B) \in \mathcal{R}$ iff $(A, B) \in Def$ and it is not the case that $B > A$.*

A preference ordering captures a notion of argument strength, and means that a defeat may not always succeed. In other words, a PAF introduces the notion of defence to represent a defeat that is not successful based on argument preferences [9].

Definition III.3. *Given $A, B \in \mathcal{A}$, an argument A defends itself against an argument B which defeats A iff A is preferred to B.*

We define two types (direct and indirect) of defence preferences as follows:

Definition III.4. *(Direct Defence Preference): Given $A, B \in \mathcal{A}$, an argument A defends itself against an argument B which defeats A iff A is directly preferred to B. We define a direct defence preference $DPref$ between arguments A and B as $DPref = A >_A B$, which means that argument A is directly preferred to argument B due to defence by A itself. For a given abstract argumentation framework (AAF) and extension $\mathcal{E}$, we denote the set of all direct defence preferences as $DPrefs = \{DPref_1, ..., DPref_n\}$.*

Definition III.5. *(Indirect Defence Preference): Given $A, B, C \in \mathcal{A}$, an argument C defends an argument A against an argument B which defeats A iff A is indirectly preferred to B because of defence by C, where A, B, C are all unique arguments. We define an indirect defence preference $IPref$ between arguments A and B as $IPref = A >_C B$, which means that argument A is indirectly preferred to argument B due to defence by a third argument C. For a given abstract argumentation framework (AAF) and extension $\mathcal{E}$, we denote the set of all indirect defence preferences as $IPrefs = \{IPref_1, ..., IPref_n\}$.*

We define a set of all defence preferences $PrefSet$ as follows.

Definition III.6. *The set of all defence preferences $PrefSet$ for a given abstract argumentation framework (AAF) and extension $\mathcal{E}$ is as follows: $PrefSet = DPrefs \cup IPrefs$, where $DPrefs$ and $IPrefs$ are the sets of direct and indirect preferences given in Definition III.4 and Definition III.5 respectively.*

IV. Computing and Verifying Preferences

We present Algorithm 3 that takes an abstract argumentation framework (AAF) and an extension (consisting of conflict-free arguments) as input and computes the set of all the defence preferences $PrefSet$ that are valid for the acceptability of the arguments in the input extension. Algorithm 1 computes the set of all direct defence preferences $DPrefs$ and Algorithm 2 computes the set of all indirect defence preferences $IPrefs$.

Algorithm 1 Compute direct defence preferences

```
Require: $AAF$, an abstract argumentation framework
Require: $\mathcal{E}$, an extension consisting of conflict-free arguments
Ensure: $DPrefs$, the set of all direct defence preferences
procedure ComputeDirectPreferences($AAF, \mathcal{E}$)
    for each $A \in \mathcal{E}$ do
        $Attackers \leftarrow \{B \mid (B, A) \in \mathcal{R}\}$    ▷ get all attackers of $A$
        for all $B \in Attackers$ do
            $Defenders \leftarrow \{C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}\}$    ▷ $C \neq A$ attacks $B$ & defends $A$
            if $Defenders = \emptyset$ then    ▷ if $B$ not attacked by any $C$
                $DPrefs \leftarrow DPrefs \cup \{A >_A B\}$
    return $DPrefs$
```

Algorithm 2 Compute indirect defence preferences

```
Require: $AAF$, an abstract argumentation framework
Require: $\mathcal{E}$, an extension consisting of conflict-free arguments
Ensure: $IPrefs$, the set of all indirect defence preferences
procedure ComputeIndirectPreferences($AAF, \mathcal{E}$)
    for each $A \in \mathcal{E}$ do
        $Attackers \leftarrow \{B \mid (B, A) \in \mathcal{R}\}$    ▷ get all attackers of $A$
        for all $B \in Attackers$ do
            $Defenders \leftarrow \{C \mid C \neq A, C \in \mathcal{E}, (C, B) \in \mathcal{R}\}$    ▷ $C \neq A$ attacks $B$ & defends $A$
            if $Defenders \neq \emptyset$ then
                for each $C \in Defenders$ do
                    $IPrefs \leftarrow IPrefs \cup \{A >_C B\}$
    return $IPrefs$
```

Algorithm 3 Compute all defence preferences

```
Require: $AAF$, an abstract argumentation framework
Require: $\mathcal{E}$, an extension consisting of conflict-free arguments
Ensure: $PrefSet$, the set of all defence preferences
procedure ComputePreferences($AAF, \mathcal{E}$)
    $DPrefs \leftarrow$ ComputeDirectPreferences($AAF, \mathcal{E}$)
    $IPrefs \leftarrow$ ComputeIndirectPreferences($AAF, \mathcal{E}$)
    $PrefSet \leftarrow DPrefs \cup IPrefs$
    return $PrefSet$
```

Algorithm 4 takes an abstract argumentation framework (AAF) and a set of all defence preferences $PrefSet$ as input and computes an extension $\mathcal{E}'$. Algorithm 5 verifies that the set of all the defence preferences $PrefSet$ returned by Algorithm 3 is correct by using Algorithm 4. If the computed extension $\mathcal{E}'$ returned by Algorithm 4 is equal to the initial extension $\mathcal{E}$ given as input to Algorithm 3, then $PrefSet$ is the correct set of all defence preferences.

The following theorem is used for verifying the correctness of the set of all defence preferences $PrefSet$.

Theorem IV.1. *Algorithm 3 is sound in that given an abstract argumentation framework AAF and an extension $\mathcal{E}$ as input, the output preference set $PrefSet$, when applied to the AAF results in the input $\mathcal{E}$ (under a given semantics).*

We now present a worked example to show the computation of preferences by using Algorithm 3 and also show how they can be verified by using Algorithm 5.

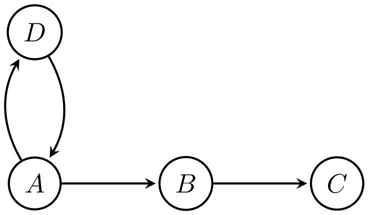

Figure 3. Example abstract argumentation framework AAF_3

Algorithm 4 Compute an extension

```
Require: $AAF$, an abstract argumentation framework
Require: $PrefSet$, the set of all defence preferences
Ensure: $\mathcal{E}$, an extension consisting of conflict-free arguments
procedure ComputeExtension($AAF$, $PrefSet$)
    for each $A \in \mathcal{A}$ do
        $Attackers \leftarrow \{B \mid (B, A) \in \mathcal{R}\}$    ▷ get all attackers of $A$
        if $Attackers \neq \emptyset$ then
            for all $B \in Attackers$ do
                $Defenders \leftarrow \{X \mid (A >_X B) \in PrefSet\}$    ▷ $A$ is preferred to $B$ since $X$ defends $A$
                if $Defenders \neq \emptyset$ then
                    $\mathcal{E} \leftarrow \mathcal{E} \cup A \cup X$
    for each $A \in \mathcal{A}$ do
        $Attackers \leftarrow \{B \mid (B, A) \in \mathcal{R}\}$    ▷ get all attackers of $A$
        $Attacked \leftarrow \{C \mid (A, C) \in \mathcal{R} \wedge C \in \mathcal{E}\}$    ▷ get arguments $A$ attacks that are in $\mathcal{E}$
        if $Attackers = \emptyset \wedge Attacked = \emptyset$ then
            $\mathcal{E} \leftarrow \mathcal{E} \cup A$
    return $\mathcal{E}$
```

Algorithm 5 Verify all defence preferences

```
Require: $AAF$, an abstract argumentation framework
Require: $\mathcal{E}$, an extension consisting of conflict-free arguments
Require: $PrefSet$, the set of all defence preferences
Ensure: $PrefSet$, the verified set of all defence preferences
procedure VerifyPreferences($AAF, \mathcal{E}, PrefSet$)
    $\mathcal{E}' \leftarrow$ ComputeExtension($AAF$, $PrefSet$)
    if ($\mathcal{E}' = \mathcal{E}$) then
        return $PrefSet$
    else
        return null
```

Example IV.1. *Let there be an abstract argumentation framework of Figure 3 and an extension $\mathcal{E}$ given as follows:*

- $\mathcal{A} = \{A, B, C, D\}$.
- $(A, B) \in \mathcal{R}$, $(B, C) \in \mathcal{R}$, $(D, A) \in \mathcal{R}, (A, D) \in \mathcal{R}$.
- $\mathcal{E} = \{A, C\}$

By using Algorithm 3, we can compute the direct defence preference $DPref = A >_A D$, since it satisfies the condition that $(D, A) \in \mathcal{R}$ and $A \in \mathcal{E}$ and $D \notin \mathcal{E}$ and there is no other argument $X \in \mathcal{E}$ that attacks D. We can compute the indirect defence preference $IPref = C >_A B$, since it satisfies the condition that $(A, B) \in \mathcal{R}$, $(B, C) \in \mathcal{R}$, $A, C \in \mathcal{E}$ and $B \notin \mathcal{E}$. Therefore, the set of all defence preferences $PrefSet = \{A >_A D, C >_A B\}$.

By using Algorithm 5, we can verify that the computed set of all defence preferences $PrefSet$ is correct. Algorithm 5 computes the extension $\mathcal{E}'$ for the given abstract argumentation framework (AAF) and set of all defence preferences $PrefSet = \{A >_A D, C >_A B\}$. Argument A is attacked by argument D but since it is preferred to D as given in PrefSet, A is added to the extension $\mathcal{E}'$, and therefore $\mathcal{E}' = \{A\}$. Argument C is attacked by argument B but since it is preferred to B as given in PrefSet, C is added to the extension $\mathcal{E}'$, and therefore $\mathcal{E}' = \{A, C\}$. Argument B is attacked by argument A but since it is not preferred to A, therefore B is not added to the extension $\mathcal{E}'$. Similarly, argument D is attacked by argument A but since it is not preferred to A, therefore D is not added to the extension $\mathcal{E}'$. The output extension $\mathcal{E}' = \{A, C\}$ is equal to the input extension $\mathcal{E}$. Thus, we conclude that the computed set of all defence preferences $PrefSet$ is correct.

V. Implementation Details

We have implemented our proposed algorithms given in Section IV for evaluation purposes, in Java and using the Tweety library [20]. Figure 4 shows the flowchart of the automated approach for preference computation and verification. Following are the steps that occur:

1) The input to the system is an abstract argumentation framework (AAF) and an extension $\mathcal{E}$ (consisting of conflict-free arguments). This is denoted by $(AAF, \mathcal{E})$.
2) The **ComputePreferences** procedure:
    - calls the **ComputeDirectPreferences** procedure (with input $(AAF, \mathcal{E})$) as given in Algorithm 1 that computes the set of direct defence preferences $DPrefs$.
    - calls the **ComputeIndirectPreferences** procedure (with input $(AAF, \mathcal{E})$) as given in Algorithm 2 that computes the set of indirect defence preferences $IPrefs$.
    - receives $DPrefs$ and $IPrefs$ from **ComputeDirectPreferences** and **ComputeIndirectPreferences** procedures respectively and computes the set of all defence preferences $PrefSet \leftarrow DPrefs \cup IPrefs$.
3) The **VerifyPreferences** procedure gets the input $(AAF, \mathcal{E}, PrefSet)$ from the **ComputePreferences** procedure and:
    - calls the **ComputeExtension** procedure to compute the extension $\mathcal{E}'$.
    - receives the extension $\mathcal{E}'$ from the **ComputeExtension** procedure and checks the condition that $\mathcal{E} = \mathcal{E}'$. If the condition is true then it returns the verified set of defence preferences $PrefSet$.

Following are some of the properties of our approach.

- An argument may be defended by more than one argument in the argumentation framework, since two ore more different arguments that defend such an argument from a defeating argument could be present in the extension of acceptable arguments.
- Moreover, an argument can defend another argument from the defeats of more than one arguments in the argumentation framework.
- An argument can defend itself from any number of arguments in the argumentation framework.
- An argument can have a direct defence preference only when it has no indirect defence preference.

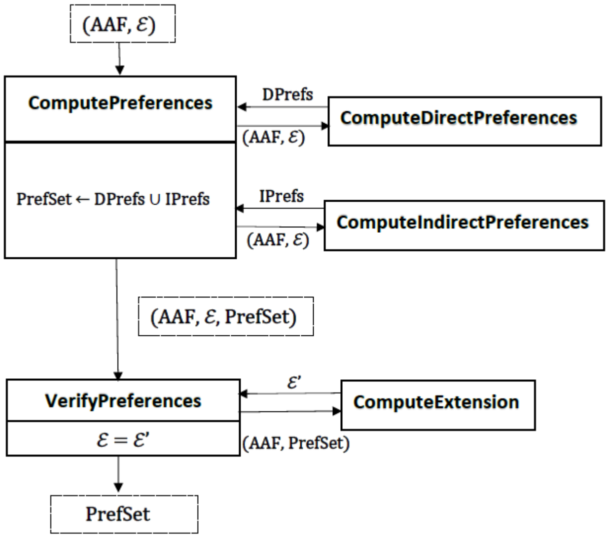

Figure 4. Flow Chart of the Automated Approach for Preference Computation and Verification

- An argument can have an indirect defence preference only when the argument defending it is present in a given extension.

VI. Conclusion

We have presented a novel approach that automates the computation and verification of preferences in an abstract argumentation system. We have implemented the algorithms and evaluated them on small abstract argumentation graphs. This provides a promising mechanism to test our computation on larger argumentation graphs to check the scalability. The novelty of our approach with respect to previous research is that:

1) Preferences are computed at the end of the argumentation process and need not be stated in advance.
2) Preferences explain the justification of the acceptability of an argument, i.e., whether an argument was able to defend itself or by another argument, from an attack. Moreover, from this information it can also be deduced why the attacking arguments were not accepted in terms of argument preferences.
3) We also provide a mechanism for verifying the computed preferences to prove their correctness.

This work has applications in decision support and recommender systems, where the resulting decision or recommendation can be explained by the preference relations.

Acknowledgements

This research was supported by EPSRC grant EP/P011829/1, *Supporting Security Policy with Effective Digital Intervention (SSPEDI)*.

References

[1] L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation. In *Proc. of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 1–7, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] L. Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *Journal of Automated Reasoning*, 29(2):125–169, Jun 2002.

[3] L. Amgoud and H. Prade. Using arguments for making and explaining decisions. *Artificial Intelligence*, 173(3):413 – 436, 2009.

[4] L. Amgoud and S. Vesic. A new approach for preference-based argumentation frameworks. *Annals of Mathematics and Artificial Intelligence*, 63(2):149–183, 2011.

[5] L. Amgoud and S. Vesic. Rich preference-based argumentation frameworks. *International Journal of Approximate Reasoning*, 55(2):585 – 606, 2014.

[6] S. Benferhat, D. Dubois, and H. Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In D. Heckerman and A. Mamdani, editors, *Uncertainty in Artificial Intelligence*, pages 411 – 419. Morgan Kaufmann, 1993.

[7] P. Besnard and A. Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1):203 – 235, 2001.

[8] B. Bonet and H. Geffner. Arguing for decisions: A qualitative model of decision making. In *Proc. of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, UAI'96, pages 98–105. Morgan Kaufmann Publishers Inc., 1996.

[9] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[10] X. Fan and F. Toni. On computing explanations in abstract argumentation. In *Proc. of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, pages 1005–1006. IOS Press, 2014.

[11] A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *Theory Pract. Log. Program.*, 4(2):95–138, Jan. 2004.

[12] S. Kaci and L. van der Torre. Preference-based argumentation: Arguments supporting multiple values. *Int. J. Approx. Reasoning*, 48(3):730–751, 2008.

[13] K. Konczak. Voting procedures with incomplete preferences. In *in Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, 2005.

[14] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artificial Intelligence*, 173(9):901 – 934, 2009.

[15] J. Muller and A. Hunter. An argumentation-based approach for decision making. In *Proc. of the 2012 IEEE 24th International Conference on Tools with Artificial Intelligence - Volume 01*, ICTAI '12, pages 564–571. IEEE Computer Society, 2012.

[16] G. Pigozzi, A. Tsoukiàs, and P. Viappiani. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence*, 77(3):361–401, 2016.

[17] M. Pini, F. Rossi, K. Venable, and T. Walsh. Incompleteness and incomparability in preference aggregation: Complexity results. *Artificial Intelligence*, 175(7):1272 – 1289, 2011.

[18] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7:25–75, 1997.

[19] G. R. Simari and R. P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(2):125 – 157, 1992.

[20] M. Thimm. The formal argumentation libraries of tweety. In *Proc. of the International Workshop on Theory and Applications of Formal Argument*, 2017.

[21] T. Walsh. Representing and reasoning with preferences. *AI Magazine*, 28(4):59–70, 2007.

[22] Q. Zhong, X. Fan, F. Toni, and X. Luo. Explaining best decisions via argumentation. In *Proc. of the European Conference on Social Intelligence (ECSI-2014), Barcelona, Spain, November 3-5, 2014.*, pages 224–237, 2014.