

The Skeptic's Argumentation Game or: Well-Founded Explanations for Mere Mortals

Shawn Bowers¹, Yilin Xia² and Bertram Ludäscher^{2,*}

¹Department of Computer Science, Gonzaga University, USA

²School of Information Sciences, University of Illinois, Urbana-Champaign, USA

Abstract

We propose a new discussion game for abstract argumentation frameworks (AFs), related to, but different from other dialectical proof procedures and discussion games: the *Skeptic's Argumentation Game* (SAG). Unlike in other AF games, Player I (the *Skeptic*) aims to establish that an argument x is defeated, while Player II (the *Optimist*) tries to prove the opponent wrong, i.e., that x is accepted. If neither player has a winning strategy, the position is a draw and x 's status is undecided. This "reversal of roles" (compared to the usual *Proponent vs Opponent* dialogue games about the acceptance status of an argument) might appear strange at first, but has a number of important, fruitful consequences. Since SAG corresponds exactly to (i) the grounded labeling semantics of AFs, and (ii) the standard semantics of "win-move" (WM) games for normal play on finite graphs, a rich body of research and results can be transferred directly to grounded AF labelings. In this paper we show one such result transfer: The value (WON/LOST/DRAWN) of a position x in a solved WM game can be fully explained by the *provenance* of x , i.e., a subgraph definable by a regular expression. Consequently, via SAG, we can provide a detailed, well-founded explanation for the acceptance status (value) of an argument. We also exploit well-founded explanations for visualization to expose the structural dependencies inherent in an AF under the grounded semantics.

Keywords

Argumentation frameworks, game theory, provenance, discussion games

1. Introduction

Dung's seminal work on abstract argumentation [1] gives a two-line logic program that specifies an *argument processing unit* (APU), i.e., a meta-interpreter that can be used to evaluate (or *solve*) an argumentation framework (AF) via a declarative semantics:

$$\begin{aligned} \text{Defeated}(x) &\leftarrow \text{Attacks}(y, x), \text{Accepted}(y). \\ \text{Accepted}(y) &\leftarrow \neg \text{Defeated}(y). \end{aligned} \quad (P_{AF2})$$

Given an AF, i.e., a digraph $G_{AF} = (V, E)$ whose edges $y \rightarrow x$ in E model that an argument $y \in V$ attacks an $x \in V$, the first rule states that an *argument* x is *defeated* if there is an *accepted* argument y that attacks x . The second rule specifies that an argument is accepted if it is not defeated. The subgoal "Accepted(y)" in the first rule can be replaced with the negated atom from the second rule, resulting in an equivalent single-rule program:

$$\text{Defeated}(x) \leftarrow \text{Attacks}(y, x), \neg \text{Defeated}(y). \quad (P_{AF1})$$

SAFA'24: 5th Intl. Workshop on Systems and Algorithms for Formal Argumentation, September 17, 2024, Hagen, Germany

*Corresponding author.

✉ bowers@gonzaga.edu (S. Bowers); yilinx2@illinois.edu (Y. Xia); ludasch@illinois.edu (B. Ludäscher)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

This rule can be written equivalently in a “reversed edges” form as follows:

$$\text{Defeated}(x) \leftarrow \text{AttackedBy}(x, y), \neg \text{Defeated}(y). \quad (P_{AF'})$$

An edge $x \rightarrow y$ in the graph then means that argument x is *attacked-by* an argument y .

Now compare $P_{AF'}$ with the single-rule “win-move” (WM) program from [2]:

$$\text{Win}(x) \leftarrow \text{Move}(x, y), \neg \text{Win}(y). \quad (P_{WM})$$

The rule P_{WM} can be seen as a *game processing unit* (GPU), i.e., a meta-interpreter that solves two-player games: A position x in the game, given by a graph G_{WM} , is (objectively) *won* if there exists a move to a position y such that the new position is *lost* for the opponent. In particular, if there are no more moves left to play¹ a position is lost.

It is easy to see that the game-solving rule P_{WM} and the AF-solving rule $P_{AF'}$ are “identical twins”, i.e., syntactic variants of the same generic query:

$$Q(x) \leftarrow E(x, y), \neg Q(y). \quad (P_Q)$$

In other words, the variants P_{WM} and $P_{AF'}$ only differ in the interpretation of E and Q .

Solving Games with WFS. It is well known that the three-valued *well-founded semantics* (WFS) of P_{WM} solves games [2]: $\text{Win}(x)$ is TRUE, FALSE, or UNDEF (*undefined*) if and only if position x is WON, LOST, or DRAWN in the game, respectively.

Solving AFs with WFS. Similarly, Dung has shown [1] that the WFS of P_{AF1} (and thus of $P_{AF'}$) yields a unique *grounded labeling* (cf. [3]): $\text{Defeated}(x)$ is TRUE, FALSE, or UNDEF if and only if x has the label OUT (*defeated*), IN (*accepted*), or UNDEC (*undecided*), respectively.

Given the syntactic correspondence of P_{WM} and $P_{AF'}$ and the fact that the WFS of P_{WM} and $P_{AF'}$ solves games and computes the grounded labeling of AFs, respectively, the following correspondences ($\text{WON} \cong \text{OUT}$, $\text{LOST} \cong \text{IN}$, $\text{DRAWN} \cong \text{UNDEC}$) are immediate:

Theorem 1 (WM-AF Correspondence). Let $G = (V, E)$ be a finite directed graph; \mathcal{M} the well-founded model of P_Q applied to the edges E ; $\lambda(x)$ the value of position x in G_{WM}^λ ; and $\text{Lab}_{AF}(x)$ the grounded label of argument x in the argumentation framework $G_{AF'}$. Then:

$$\mathcal{M}(Q(x)) = \begin{Bmatrix} \text{TRUE} \\ \text{FALSE} \\ \text{UNDEF} \end{Bmatrix} \Leftrightarrow \lambda(x) = \begin{Bmatrix} \text{WON} \\ \text{LOST} \\ \text{DRAWN} \end{Bmatrix} \Leftrightarrow \text{Lab}_{AF}(x) = \begin{Bmatrix} \text{OUT} \\ \text{IN} \\ \text{UNDEC} \end{Bmatrix}$$

Here, G_{WM}^λ denotes the solved (= *labeled*) game graph (see Definition 3 for details) and $\text{Lab}_{AF}(x)$ denotes the label (see Definition 13) of argument x under the grounded semantics.

Research Questions. This WM-AF correspondence raises a number of interesting issues: What is the nature of this game G_{WM} that we can play on the (reversed) AF graph $G_{AF'}$? At first glance, it seems rather counter-intuitive that an OUT (*defeated*) argument x in AF should correspond to a WON position in G_{WM} ! On the other hand, the simple (and “standard”) form

¹In chess: *Checkmate*!

of G_{WM} suggests that many classic results from game theory [4, 5, 6] carry over to abstract argumentation, thereby opening up opportunities for cross-fertilization and new insights. For example, how does G_{WM} relate to the results about the various discussion games [7] that are well-known in formal argumentation?

Outline and Contributions. In Section 2, we recall notions of impartial two-player games on finite graphs and introduce our running example: Figure 1 depicts a simple win-move game G_{WM} , its well-founded, labeled *solution* G_{WM}^λ (colors encode node labels), and—importantly—the solved game with additional *provenance* information G_{WM}^λ . Our provenance model for games is presented and can be used to precisely explain why and how the value of a position in a game depends on the values of other positions. The model builds on prior work on provenance and games from database theory (including our own [8, 9]) and includes: (i) a novel classification of provenance edges; and (ii) an elegant and efficient query mechanism based on *regular path queries* (RPQs) [10] to extract the provenance of a node.

In Section 3, we solve the puzzle from above and present (iii) the *Skeptic’s Argumentation Game* (SAG), a new discussion game for AF implied by the WM-AF correspondence described above. Via the duality of WM and AF, the results developed for game provenance (Section 2) carry over to abstract argumentation and allow us to (independently) rediscover well-established notions and results from AF (demonstrating the robustness of these AF notions). For example, the concept of *min-max numbering* of arguments, known from *strongly admissible sets* [11], is closely related to the notion of *length* (of a position) in game theory. Position lengths, in turn, are useful for explaining why an argument is IN, OUT, or UNDEC in the grounded labeling, yielding (iv) an alternative approach for *explaining the acceptance status* of arguments.

In Section 4, we present an application facilitated by our game-theoretic, provenance-aware approach to grounded AF solutions, notably (v) a *layered visualization* approach, illustrated using a well-known example from case law [12]. A prototypical implementation of our approach using a Jupyter notebook is available [13] and has been integrated into the PyArg system [14].

2. Games and their Provenance

We begin by briefly describing win-move games and algorithms for solving them. We then discuss *game provenance* [9], its regular structure (leading to a classification of edge types), and techniques for using provenance to explain the value of a position in a solved game.

2.1. Games and Position Values

Consider the graph G_{WM} in Figure 1a, which will serve as our running example. With such a graph we can associate a “classic” (i.e., impartial two-player) game as follows.

Definition 1 (Game). A (win-move) *game* is a finite digraph $G_{WM} = (V, E)$ where V is a set of *positions*, and edges $(x, y) \in E \subseteq V \times V$ (also denoted $x \rightarrow y$) represent possible moves.

After agreeing on a starting position $x_0 \in V$, the game is played with a pebble by two players who take turns moving in *rounds* (a round consists of two *moves*). Player I starts from x_0 . A player can move from position x to y if $(x, y) \in E$. In this case, y is a *follower* of x .

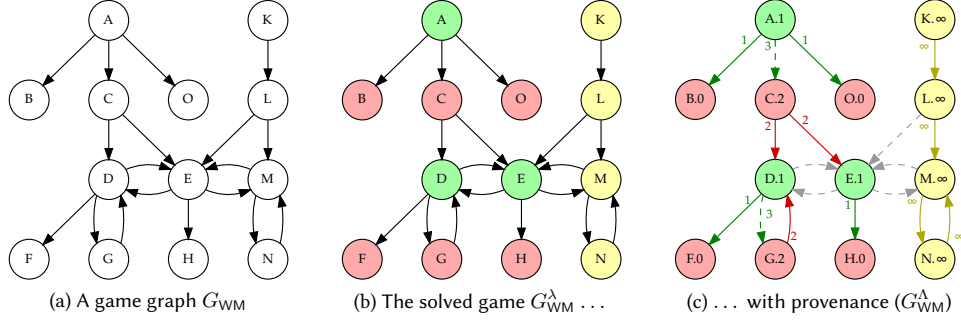


Figure 1: Solving the game in (a) yields G_{WM}^λ in (b): node colors green, red, and yellow represent values WON, LOST, and DRAWN, respectively. Additional bookkeeping, while solving the game, yields (c): G_{WM}^λ , a solved game *with provenance* explaining, e.g., optimal moves, and how quickly a win can be forced (or how long a loss can be delayed).

A *play* π is an alternating sequence of moves $x_0 \xrightarrow{I} x_1 \xrightarrow{II} x_2 \xrightarrow{I} \dots$ by the players. The *length* $|\pi|$ of a play is the length of the sequence. A play π is *complete* if either $|\pi| = \infty$ (repeating moves²) or π ends after $|\pi| = n$ moves in a terminal node. A player who cannot move *loses* (so-called *normal play*) and the opponent *wins*. If $|\pi| = \infty$ the play is a *draw*.

Example 1. If Player I starts from position D in Figure 1a, the optimal move is to F, thereby leaving Player II stranded in a terminal node. However, the move $D \rightarrow E$ by Player I is a *blunder*: it leaves Player II in a winning position (via the move $E \rightarrow H$). Alternatively, if Player I starts from C, regardless of their next move (which is either to D or to E), Player II can win the game.

Intuitively, the (objective!) *value* of a position (WON, LOST, or DRAWN) only depends on “best moves” (optimal play). In particular, “bad moves” (*blunders*) do not affect the value of a position.

Definition 2 (Value of a Position). A position x is *WON* if a player can force a win from x , independent of the opponent’s moves; x is *LOST* if there are no moves to play or if the opponent can force a win; and x is *DRAWN* if neither player can force a win.

A player that can force a win from a position has a *winning strategy*: a set of moves they can make that lead to a loss for their opponent regardless of the moves their opponent makes.

Example 2. The winning strategy for a player at E in Figure 1a consists of a single move $\{E \rightarrow H\}$. The other moves from E are blunders. There are three winning strategies for a player starting at A: $\{A \rightarrow B\}$; $\{A \rightarrow O\}$; and $\{A \rightarrow C, D \rightarrow F, E \rightarrow H\}$ (depending on whether the opponent moves to D or E, respectively).

Determining the value of each position results in a *solved game*, which is represented via a total labeling function λ .

Definition 3 (Solution Labeling of a Game). The (solution) labeling $\lambda : V \rightarrow \{\mathbf{W}, \mathbf{L}, \mathbf{D}\}$ of $G_{WM} = (V, E)$ yields the position values, where \mathbf{W} , \mathbf{L} , and \mathbf{D} are shorthand for WON, LOST, and DRAWN positions, respectively. The corresponding *solved game* is denoted $G_{WM}^\lambda = (V, E, \lambda)$.

²On finite graphs, cycles are necessary (but not sufficient) for a play to end in a draw.

Figure 1b shows the labeled solution of the game in Figure 1a (node colors indicate labels).

2.2. Standard Algorithms for Solving Games

It is well known that games can be solved by iterating the following two rules³.

- $\lambda(x) := \mathbf{L}$ if $\forall (x, y) \in E: \lambda(y) = \mathbf{W}$. (RR)
- $\lambda(x) := \mathbf{W}$ if $\exists (x, y) \in E: \lambda(y) = \mathbf{L}$. (GR)

The *red rule* (RR) states that a position x is **LOST** (labeled **L**) if *all* of x 's followers y have already been **WON** (labeled **W**): No matter which follower y of x a player moves to, the opponent can force a win from y . The *green rule* (GR) states that a position x is **WON** (labeled **W**) if *at least one* of x 's followers y has already been **LOST** (labeled **L**): A player can thus choose to move from x to such a y , leaving the opponent in a lost position.

On an unlabeled graph G_{WM} , the first applicable rule is RR: Terminal positions have no moves and so the RR \forall -condition is vacuously true. The result is that each such terminal position is assigned **L**. In the next iteration, GR becomes applicable, assigning **W** to all positions that have at least one direct move according to the GR \exists -condition to a terminal position. The second iteration of RR labels positions whose moves all lead to positions previously won. The second iteration of GR then assigns labels to positions with at least one move to a lost position. This stage-wise iteration eventually converges to a *fixpoint*, and any remaining nodes are drawn positions [9, 15]. The underlying process of iterating RR and GR is equivalent to the classic *backward induction* procedural approach for solving games.

A game can also be solved by evaluating the rule P_{WM} under the *well-founded semantics* [2]. If this is implemented via the *alternating fixpoint procedure* (AFP) [16], one obtains an increasing sequence of underestimates $U_1 \subseteq U_3 \subseteq U_5 \dots$ converging to the set of **TRUE** atoms U^ω from below, and a decreasing sequence of overestimates $O_0 \supseteq O_2 \supseteq O_4 \dots$ converging to O^ω , the **TRUE** or **UNDEF** (undefined) atoms from above. Thus, the atoms in the “gap” $O^\omega \setminus U^\omega$ have the third truth-value **UNDEF**, while atoms not in O^ω are **FALSE**.

2.3. Explaining Position Values through Provenance

The *provenance* $\mathcal{P}(x)$ represents an explanation of why and how a position x has a particular value in a (solved) game. To compute $\mathcal{P}(x)$, we first add additional provenance information, which includes position lengths and edge labels, to solved games. We then use the provenance information to construct the explanation $\mathcal{P}(x)$ as a subgraph of the solved game rooted at x (together with the additional provenance information). Here we consider two notions of provenance: the *actual* [9] and *primary* [17] provenance of a position.

Solving games using either AFP or (equivalently) backward induction yields additional game-theoretic information, notably the *length* (or *remoteness* [4]) of a position.

Definition 4 (Position Length). Let $G_{WM}^\lambda = (V, E, \lambda)$ be a solved game. The *length* $|x|$ of a position $x \in V$ is: the minimum number of moves necessary to force x 's win if $\lambda(x) = \mathbf{W}$ (i.e., x 's value becomes known after its first follower is **LOST**); the maximum number of moves that

³Initially, $\lambda(x) := \emptyset$ for each position $x \in V$.

x 's losing can be delayed if $\lambda(x) = \mathbf{L}$ (i.e., x 's value is known after its last follower is won); and ∞ (denoting infinite play) if $\lambda(x) = \mathbf{D}$.

Position length corresponds to the classic game-theoretic notion of *optimal play*: players try to win as quickly or lose as slowly as possible thus avoiding blunders. The length of a position can be computed simply by using the iteration/state number in which the position's value becomes first known (starting at 0) [9, 17].

Definition 5 (Provenance Move Labels). Let $G_{\text{WM}}^\lambda = (V, E, \lambda)$ be a solved game. A *provenance move labeling* $\Lambda : E \rightarrow \{\mathbf{w}, \mathbf{l}, \mathbf{d}, \mathbf{b}\}$ assigns labels for *winning* (\mathbf{w}), *delaying* (\mathbf{l}), *drawing* (\mathbf{d}), and *blundering* (\mathbf{b}) moves, such that:

$$\Lambda(x, y) := \begin{cases} \mathbf{w} & \text{if } \lambda(x) = \mathbf{W} \text{ and } \lambda(y) = \mathbf{L} \\ \mathbf{l} & \text{if } \lambda(x) = \mathbf{L} \text{ and } \lambda(y) = \mathbf{W} \\ \mathbf{d} & \text{if } \lambda(x) = \mathbf{D} \text{ and } \lambda(y) = \mathbf{D} \\ \mathbf{b} & \text{otherwise} \end{cases}$$

A solved game with move labels is denoted $G_{\text{WM}}^\Lambda = (V, E, \lambda, \Lambda)$.

Not all moves are created equal. The previous definition assigns four different edge types (labels) $\Lambda(x, y)$ to moves $(x, y) \in E$. The edge type depends on the value $\lambda(x)$ of the move's origin x and the value $\lambda(y)$ of its follower position y . Table 1 provides an overview: If a position x is won, there must be a *winning* move to a follower y that is lost for the opponent. Choosing any other follower (i.e., one that is drawn or won for the opponent) is a *blunder*. On the other hand, if x is drawn, there cannot be a lost follower (otherwise x would be winning, not drawn). Instead one must find a drawn follower y to keep the draw. Moving to a follower that is won (for the opponent) is another kind of *blunder*. Finally, in a lost position x , there are no lost or drawn followers (otherwise, x would not be lost) and the only option is a *delaying* move to a position y that is won for the opponent.

From x To y	LOST (\mathbf{L})	DRAWN (\mathbf{D})	WON (\mathbf{W})
WON (\mathbf{W})	winning (\mathbf{w})	<i>blunder</i> (\mathbf{b})	<i>blunder</i> (\mathbf{b})
DRAWN (\mathbf{D})	—	drawing (\mathbf{d})	<i>blunder</i> (\mathbf{b})
LOST (\mathbf{L})	—	—	delaying (\mathbf{l})

Table 1

Move labels (edge types) for moves $x \rightarrow y$ from Definition 5: provenance-relevant edge types (\mathbf{w} , \mathbf{d} , \mathbf{l}) are found on the diagonal; there are three types of blunders and three nonexistent edge types.

Like positions, moves in a solved game are also associated with a *length*:

Definition 6 (Move Length). Let $G_{\text{WM}}^\Lambda = (V, E, \lambda, \Lambda)$ be a solved game with move labels. The *length* $|x, y|$ of a move $(x, y) \in E$ is: $1 + |y|$ if $\Lambda(x, y) \in \{\mathbf{w}, \mathbf{l}\}$; ∞ if $\Lambda(x, y) = \mathbf{d}$; and undefined if $\Lambda(x, y) = \mathbf{b}$.

Move labels (edge types) are used directly to define the *actual provenance* $\mathcal{P}_{\text{ac}}(x)$ of a position x in a solved game [17].

Definition 7 (Actual Provenance of a Position). Given a provenance-labeled solution G_{WM}^Λ , the *actual provenance* $\mathcal{P}_{ac}(x)$ of a position x is the subgraph reachable from x by only following w -, l -, and d -labeled moves. In particular, blundering (b) moves must be ignored.

Not all winning moves are created equal. Winning moves can be further categorized as either *primary* or *secondary* based on optimal play. A *primary winning move* (labeled w_{pr}) from x is a winning move that is part of a shortest-length win for x . A *secondary winning move* (labeled w_{sc}) from x is a non-shortest winning move. Both the AFP-based and backward-induction algorithms for solving games can be instrumented to compute all primary winning (w_{pr}) and delaying (l) edge labels [17]. The remaining edge labels can be obtained using Definition 5.

Definition 8 (Primary Provenance of a Position). Given a provenance-labeled solution G_{WM}^Λ , the *primary provenance* $\mathcal{P}_{pr}(x)$ of a position x is the subgraph reachable from x by only following w_{pr} -, l -, and d -labeled edges. Thus, blundering (b) and secondary winning moves (w_{sc}) are ignored.

Example 3. Figure 1c shows the solved game of Figure 1b with corresponding provenance move labels and lengths. Edge colors indicate corresponding labels, where primary winning moves are denoted using solid edges, secondary winning moves are denoted using dashed edges, and blundering moves are drawn using gray dashed edges.

2.4. The Regular Structure of Game Provenance

The type graph in Figure 3a below summarizes the overall provenance structure of solved games with respect to position values and move types. The seven edge types can be split into provenance-relevant moves (*winning*, *delaying*, *drawing*) and provenance-irrelevant moves (three types of *blunders*). The *winning* moves can be further subdivided into *fast-winning* (primary provenance) and *slow-winning* (secondary provenance). Figure 3a (like Table 1) also shows that there are three types of “ghost moves” that cannot exist in a solved game.

For example, a position would not be lost if there were a move to a lost or to a drawn position. Similarly, a drawn x can never have a move to a lost follower y , otherwise x would be winning rather than being drawn.

Computing Provenance using RPQs. Obtaining the provenance of a position in a solved, provenance-labeled game can be succinctly expressed using *regular-path queries* (RPQs) [10] (i.e., over the type graph of Figure 3a). Consider a solved game $G_{WM}^\Lambda = (V, E, \lambda, \Lambda)$. Its move relation E (with labelings) induces a function M^R (for *move paths matching* R). Here R is a regular expression over an alphabet $\{w_{pr}, w_{sc}, w, l, d, b\}$ of move labels.

Definition 9 (Move-Based Regular Path Queries). The expression $M^R(x)$ evaluates to the minimal subgraph $G' \subseteq G$ rooted at x such that all paths

$$x \xrightarrow{\ell_1} x_1 \xrightarrow{\ell_2} x_2 \cdots \xrightarrow{\ell_n} x_n$$

in G , whose concatenated labels $\ell_1 \ell_2 \cdots \ell_n$ match regular expression R , also match in G' .

In this way, the parameter R specifies a path expression, but unlike an RPQ which returns a set of nodes, $M^R(x)$ returns a subgraph definable by an RPQ R .

Theorem 2 (Provenance via RPQs). The actual $\mathcal{P}_{ac}(x)$ and potential $\mathcal{P}_{pr}(x)$ provenance of a position x can be computed using M^R (where w denotes $w_{pr}|w_{sc}$):

$$\mathcal{P}_{ac}(x) := \begin{cases} M^{w \cdot (l \cdot w)^*}(x) & \text{if } \lambda(x) = \bar{W} \\ M^{(l \cdot w)^*}(x) & \text{if } \lambda(x) = \bar{L} \\ M^{d^+}(x) & \text{if } \lambda(x) = \bar{D} \end{cases}$$

$$\mathcal{P}_{pr}(x) := \begin{cases} M^{w_{pr} \cdot (l \cdot w_{pr})^*}(x) & \text{if } \lambda(x) = \bar{W} \\ M^{(l \cdot w_{pr})^*}(x) & \text{if } \lambda(x) = \bar{L} \\ M^{d^+}(x) & \text{if } \lambda(x) = \bar{D} \end{cases}$$

Finally, we note that M^R -style expressions can be used to define additional (path-based) queries over a game graph (e.g., to check reachability between nodes, to find blundering paths, or to only consider secondary provenance).

3. The Skeptic's Argumentation Game

We briefly recall some basic definitions of abstract argumentation frameworks [1, 18] and then introduce SAG, the *Skeptic's Argumentation Game*.

3.1. Preliminaries: Abstract Argumentation Frameworks

Definition 10. An *argumentation framework* (AF) is a finite digraph $G_{AF} = (V, E)$ where the nodes V represent *arguments* and edges $(x, y) \in E \subseteq V \times V$ (denoted $x \rightarrow y$) represent *attacks*.

Within an AF, a set S of arguments *attacks* y if y is attacked by some argument $x \in S$.

Definition 11. Let $G_{AF} = (V, E)$. $S \subseteq V$ is *conflict free* if no two arguments in S attack each other. An argument x is *acceptable* for S if every argument y that attacks x is attacked by S (S is said to *defend* x). S is *admissible* if it is conflict free and each argument in S is acceptable with respect to S .

An AF can have many admissible sets, referred to as *extensions*. Different classes of extensions give rise to different *extension semantics*.

Definition 12. For $G_{AF} = (V, E)$, $S \subseteq V$ is *strongly admissible* if every argument in S is defended by some subset $S' \subseteq S \setminus \{x\}$ such that S' is also strongly admissible.

An AF can also have many strongly admissible sets, the largest of which is the unique *grounded extension* [19, 11]. Dung [1] showed that the well-founded model of P_{AF2} exactly gives an AF's grounded extension.

Given an AF, we can consider its extensions as the “solutions” (under the given extension semantics) for that AF. An alternative approach is to *label* the arguments according to their *acceptance status* [20, 11]:

Definition 13. (Labeling Semantics) Let $G_{AF} = (V, E)$ be an argumentation framework. An AF *labeling* is a function $Lab_{AF} : V \rightarrow \{\text{IN}, \text{OUT}, \text{UNDEC}\}$ that assigns a label of IN, OUT,

or UNDEC to arguments. Label values correspond to whether an argument is IN an extension (*accepted*), OUT of an extension (*rejected*), or neither in nor out (UNDEC).

3.2. Discussion Games

The *standard grounded game* (SGG) and the *grounded discussion game* (GDG) are alternative game-based proof procedures for determining whether an argument is in the grounded extension [11].

The Standard Grounded Game (SGG). The SGG defines a *discussion* (similar to a play) as a sequence of arguments (agreeing with the given attack relation) made by two players, the *Proponent* (trying to show that an argument is *accepted*) and the *Opponent* (trying to refute this claim). Unlike the traditional game-theoretic set-up described in Section 2, where Player I makes the first move from the start node x to a follower y , in SGG, it is the Opponent who first chooses a move to a successor node y (i.e., a follower in the attacked-by direction of edges).

The Grounded Discussion Game (GDG). The GDG is played by two players directly on an AF graph (not on a game tree) and players have four types of moves at their disposal: (i) claim an argument x is labeled IN (HTB(x): *has to be the case*); (ii) claim an argument should not be labeled OUT (CB(x): *can be the case*); (iii) *concede* that an argument is labeled IN (CONCEDE(x)); or (iv) *retract* that an argument should not be labeled OUT (RETRACT(x)).

SGG and GDG Compared. One could argue that SGG is conceptually simpler than GDG since the latter introduces various move types (HTB, CB, CONCEDE, and RETRACT).

On the other hand, SGG is computationally more expensive, as “the number of steps needed in a winning strategy of the SGG can be *exponential* in relation to the in/out-size of the strongly admissible labeling that the SGG winning strategy is constructing” [11, p.297].

In the following, we propose the *Skeptic's Argumentation Game* (SAG), which can be seen as a variant of the SGG (e.g., w.r.t. its conceptual simplicity) but without its exponential overhead. It combines the benefits of both approaches while also following a traditional game-theoretic set-up. In a sense, SAG is not a new game but rather a new interpretation (for grounded AF semantics) of the classic win-move game described in Section 2.

3.3. The Skeptic's Argumentation Game

Figure 2a shows an AF with its corresponding grounded labeling in Figure 2b and provenance information in Figure 2c, where IN (*accepted*) is **blue**, OUT (*defeated*) is **orange**, and UNDEC (*undecided*) is **yellow**. Note that Figure 2 is equivalent to Figure 1 as follows:

- (a) the WM-graph in Figure 1a is identical to the AF-graph in Figure 2a if we reverse the *attack* edges (i.e., replace them by *attacked-by* edges); and
- (b) the node values in Figure 1b and Figure 1c match those in Figure 2b and Figure 2c, respectively, see Theorem 1 (WM-AF Correspondence).

As suggested in Section 1, this correspondence (or *duality*) implies that the reverse of the attack graph (i.e., the *attacked-by* graph) can be understood as a 2-player (argumentation) game that is isomorphic to the standard WM game. This allows us to directly transfer the provenance results from Section 2 to the AF setting (e.g., see the example in Figure 2c). But what is this game?

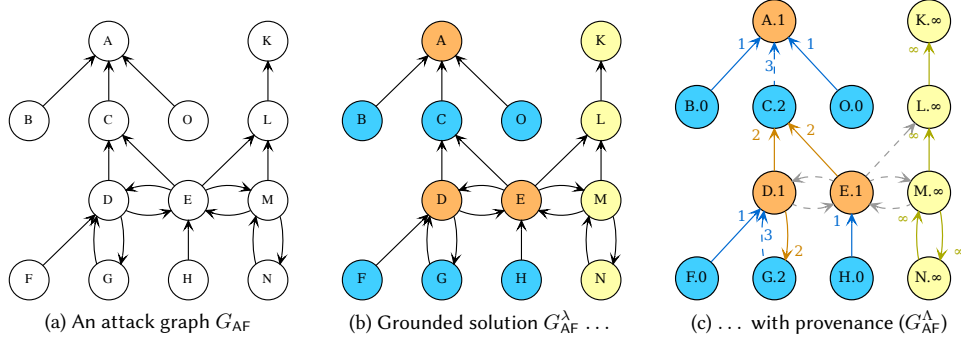


Figure 2: Solving the AF in (a) yields G_{AF}^λ in (b): node colors orange, blue, and yellow represent labels OUT, IN, and UNDEC, respectively. Additional bookkeeping, while computing the grounded labeling of AF, yields (c): G_{AF}^λ , a grounded labeling with provenance explaining, e.g., how quickly a SAG position can be won (or how long a loss can be delayed).

A Skeptic’s Perspective. Player I, the *Skeptic*, argues that a node x in AF is a *defeated* argument. To this end, the Skeptic claims that there *exists* an attacker y in the AF-graph which itself is accepted and y attacks x (or equivalently x is *attacked-by* y). This corresponds to Player I choosing one of the possible moves (from x) in the game. Player II, the *Optimist*, begs to differ and makes the counter claim that *all* attackers of x are defeated, including y .

Example 4 (A Skeptic’s Win). Assume the Skeptic wants to demonstrate that node D is a defeated argument in Figure 2a. The best move in SAG is to move to node F (which attacks D) and claim that F is accepted and thus D is defeated. The Optimist could try and show that F itself is defeated (so the attack $F \rightarrow D$ wouldn’t matter), alas there are no further moves to play: No argument in AF attacks F, so its acceptance has been established and the Skeptic has proven that D is indeed defeated.

Example 5 (An Optimist’s Win). Alternatively, if the Skeptic claims that C is defeated, there must exist an accepted attacker of C. Both possible SAG moves, i.e., to the attacker D and to the attacker E can be refuted by the Optimist: D is defeated (as just shown in Example 4) so its attack on C can be ignored. Argument E is similarly defeated (via the accepted attacker H). Since *all* attackers of C have been shown to be defeated, the Skeptic’s claim that C is defeated has been refuted and C must be accepted: A win for the Optimist (Player II).

Looking back at Example 4, had the Skeptic chosen the move from D to E (instead of to F), this would have been a *blunder*: the Skeptic aims to point to an accepted attacker of D but the potential attacker E is itself defeated, so this attack cannot be used to establish the defeat of D.

As shown for (generic) win-move games in Section 2.4, one can distinguish at least seven different kinds of edges, i.e., *not all moves are created equal*: winning, losing (i.e., delaying), and drawing moves are the only provenance edges that contribute to the value of a position x (WON, LOST, or DRAWN). In contrast, blunders are moves that are not considered part of the provenance of x , as they do not contribute to x ’s value.

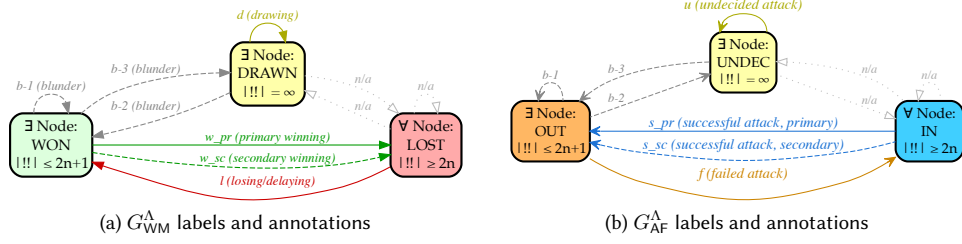


Figure 3: Provenance Edge Types. (a) Node x is a won if there exists (\exists) a (primary or secondary winning) move to a LOST position y . Node x is LOST if all (\forall) moves lead to WON positions (for the opponent). The length of x indicates how long the game lasts in optimal play: A win can be forced from a won node in at most $2n+1$ moves; a loss can be delayed for at least $2n$ moves. (“!!!” denotes a brilliant move in chess.) If neither player can force a win from x , then that node is DRAWN: There exists (\exists) a move that keeps the draw (by eventually repeating moves and playing a game of infinite length). Edge types “ $b-1$, $b-2$, $b-3$ ” are blunders and thus irrelevant for determining the value of a node. The label “ n/a ” indicates move types that cannot exist in a game. The dual interpretation in (b) is explained in the text.

SAG: A New (Old) Impartial Game. The Skeptic’s Argumentation Game (SAG) introduced above (Examples 4 and 5) employs the standard machinery of games as defined in Section 2. SAG therefore *is* such a classic impartial game⁴, only that moves are interpreted as *attacked-by* edges. Consequently, via the WM-AF correspondence, the definitions of games, solutions, the AFP-based (backward induction) algorithm, length, actual and primary provenance, and the regular provenance structure all immediately apply to this “new” argumentation game as well.

The Regular Structure of AF Provenance. Similar to Figure 3a, the *type graph* in Figure 3b summarizes the overall provenance structure of AFs with respect to argument values and attack types. The seven attack types are divided into provenance-relevant attacks (*successful*, *failed*, *undecided*) and provenance-irrelevant attacks (three kinds of *blunders*). The *successful* attacks are further subdivided into *primary* and *secondary* attacks (as in Figure 3a). Figure 3b also shows the three edge types that cannot exist in a grounded labeling. For example, an argument would not be accepted if it were attacked by an accepted or an undecided argument. Similarly, an undecided argument x can never be attacked by an accepted argument y , otherwise x would be defeated rather than undecided.

4. Provenance-Aware Layered AF Visualizations

Graph-based visualization can help to clarify the structural context of both games and argumentation frameworks, providing insight into why a node (position or argument) has a particular value. A direct result of the duality between games and AFs is that game provenance can be exploited within AF visualizations to organize arguments within attack graphs according to the role they play in the grounded labeling. Here we show one approach for exploiting this structure in which provenance edge types are used to generate AF graph layouts that are organized into

⁴Using the usual *normal play* rule, i.e., the player making the last move wins.

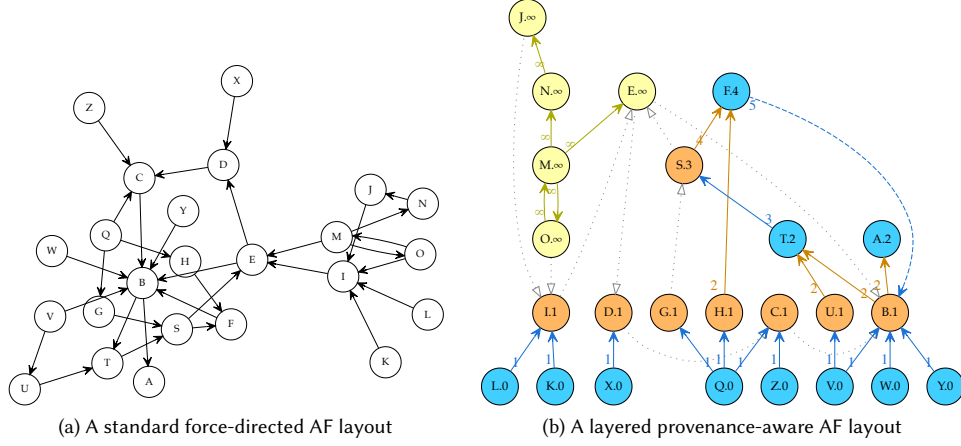


Figure 4: Provenance-aware AF visualization: (a) G_{AF} for a body of case law [12]. Even for fairly small graphs it is difficult to spot arguments that are immediately accepted, i.e., nodes without incoming attack edges. The provenance-aware, layered rendering in (b) exposes the grounded (well-founded) labeling: accepted arguments without attackers are layered at the bottom ($len = 0$), followed by immediately defeated arguments ($len = 1$), etc. The undecided component ($len = \infty$) is in a separate cluster; the most “difficult” accepted argument ($len = 4$) is F. Edges that do not contribute to provenance (gray, dotted) are suppressed, revealing the underlying structure and “dependency flow” of the solved AF.

“well-founded layers”: argument clusters that are layered according to the order their values become known in the iterative (AFP-based) computation used to compute the grounded labeling.

Example 6 (Bench-Capon Animal Case-Law). Figure 4a shows the AF from [12] visualized with a widely-used force-directed layout [21] (in this case, implemented in Graphviz⁵). Figure 4a only shows the input AF and has no additional provenance information. However, even with additional coloring and annotations, it would be difficult to quickly understand the arguments and attacks that explain a given argument’s value. As an example, T is accepted under the grounded semantics, however, determining why this is the case is difficult to untangle from the visualization in Figure 4a. Figure 4b shows the provenance-aware layered visualization of the same AF under the grounded semantics, where arguments are clustered according to their lengths. Given an argument, the relevant subgraph that led to its value is shown “below” it in the layout. For example, we can see that T is accepted because it is attacked by U and B, which are themselves attacked by the unattacked arguments V, W, and Y.

4.1. AF Layered Visualization Heuristics

A visualization of an AF using our layered approach is defined by four heuristics:

- (i) *Arguments and Attacks.* Each argument and attack is decorated according to the template of Figure 3b, following the general style of Figure 2c.

⁵<https://graphviz.org/>

- (ii) *Layout Direction*. AFs are displayed according to a *layout direction* (w.r.t. attack edges), which is either vertical or horizontal. Vertical layouts can be bottom-to-top (as shown in Figure 4b) or top-to-bottom. Horizontal layouts can be left-to-right or right-to-left.
- (iii) *Layered Arguments*. All arguments with the same length, except for those that are UNDEC (with a length of ∞), are displayed at the same level in the graph. Each level is drawn orthogonal to the (either horizontal or vertical) layout direction. This means that arguments with length k are aligned horizontally for a vertical layout, and vertically for a horizontal layout. Levels are displayed in order according to their lengths, e.g., level k in a bottom-to-top layout will be below level $k + 1$, and so on. There are five levels shown in Figure 4b: the top-most level contains F (with length 4), followed by a level containing S (with length 3), a level containing the arguments with length 2, a level containing the arguments with length 1, and the last level containing the arguments with length 0.
- (iv) *Layered Edges*. All attacks $x \rightarrow y$ that are labeled as either primary succeeding or else failing (see Figure 3b) are required to be displayed in the layout direction (e.g., “up-the-page” in a bottom-to-top layout). These edges represent the attacks that are relevant in the explanation of an argument’s value, as opposed to irrelevant (blunder) attacks or secondary (successful) attacks discovered after an argument’s value has already been found in the backward induction (AFP-based) computation. Additional edges are drawn as appropriate for the graph, however, they do not impact the overall layering (i.e., they are not required to follow the same layout direction constraints).

Example 7. In Figure 4b, the successful attack of H on F is a primary edge, and thus, is required to be drawn in the upward direction in the (bottom-to-top) layout. However, the attack of F on B is not required to be drawn according to the layered edge constraint since F’s value was known prior to considering the attack in the AFP-based solution (i.e., the attack is a secondary edge). The attack of D on C is irrelevant (a blunder), and in this case is drawn horizontally, since both arguments are in the same layer (i.e., they have the same length).

4.2. Implementing AF Layered Visualization using Graphviz in PyArg

Given an AF and its associated provenance labelings, the above heuristics can be directly implemented using the DOT language of Graphviz [22]. In particular, argument nodes and attack edges are represented according to their provenance labelings (to assign labels, colors, solid vs dashed renderings, and arrow heads). Nodes with non-infinite lengths are grouped using DOT’s rank command. Non-primary and non-failing edges are assigned the “non-ranking” constraint (i.e., the edge is ignored by Graphviz when calculating the layout of nodes). An initial prototype [14] has been implemented within PyArg [23], which also includes additional options for structuring AF visualizations under the grounded semantics using provenance information.⁶

⁶Available at: <https://pyarg.npai.science.uu.nl/21-visualise-abstract>

5. Summary and Future Work

We have explored the duality between the win-move rule P_{WM} and the AF rule P_{AF} under the well-founded semantics, giving rise to solved two-player combinatorial games and grounded AF labelings, respectively. A discussion game has been presented which, to the best of our knowledge, has not been studied by the argumentation community until now: In the *Skeptic's Argumentation Game* (SAG), Player I, the Skeptic, aims to establish that an argument x is defeated, while Player II, the Optimist, tries to prove the opponent wrong and show that x is accepted. This reversal of roles (from the more common *Proponent vs Opponent* in other discussion games) appears counter-intuitive at first, but yields a number of important results. SAG is identical to a classic/generic impartial game (only that moves are interpreted as *attacked-by* edges). Therefore, concepts and results from game theory directly carry over to AFs, e.g., backward induction, length of positions, and prior work on the provenance of games [17, 9].

A new, fine-grained classification of attack types has been derived from the dual edge types in our prior work on game provenance [17]. While many approaches have been developed to explain acceptance and non-acceptance of arguments in AFs under different semantics (e.g., see [11, 24, 25, 26]), our work complements these approaches (for the grounded semantics) via a new AF attack-type classification. Our approach also facilitates the use of path-based queries (RPQs) over AF-graphs to obtain, e.g., the *actual* and *primary* provenance of *accepted*, *defeated*, and *undecided* arguments. RPQs over provenance edge types can also provide a flexible mechanism for expressing ad-hoc queries for exploring and extracting information from AFs.

Finally, we have described an application of game provenance for AFs that leverages both edge types and node lengths to visualize the well-founded explanations of an argument's value (acceptance status). By both coloring arguments and edges according to their acceptance status, and clustering argument nodes into layers according to when their values are discovered in a backward induction, the overall “flow of acceptance and defeat” is emphasized. This approach suggests that the dependencies of arguments can be more easily discovered within an AF visualization (e.g., when compared to force-directed or other non-semantic layout approaches).

In future work, we plan to continue to explore the duality and deep connections between win-move games (and game-theoretic approaches in general) and argumentation frameworks started recently [15]. We hope that others will join in this quest for new insights at the intersection of argumentation and game theory.

References

- [1] P. M. Dung, On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, *Logic Programming and n-Person Games*, AI 77 (1995) 321–357.
- [2] A. Van Gelder, K. A. Ross, J. S. Schlipf, The Well-founded Semantics for General Logic Programs, *Journal of the ACM* 38 (1991) 619–649.
- [3] P. Baroni, M. Caminada, M. Giacomin, Abstract Argumentation Frameworks and Their Semantics, in: [18], 2018, pp. 159–236.
- [4] C. Smith, Graphs and Composite Games, *Journal of Combinatorial Theory* 1 (1966) 51–81.

- [5] A. Fraenkel, Combinatorial Game Theory Foundations Applied to Digraph Kernels, *Electronic Journal of Combinatorics* 4 (1997) 1–17.
- [6] J. Flum, Games, Kernels, and Antitone Operations, *Order* 17 (2000) 61–73.
- [7] M. Caminada, Argumentation Semantics as Formal Discussion, in: [18], 2018, pp. 487–518.
- [8] J. Flum, M. Kubierschky, B. Ludäscher, Total and Partial Well-Founded Datalog Coincide, in: *Intl. Conf. on Database Theory (ICDT)*, LNCS 1186, Springer, 1997, pp. 113–124.
- [9] S. Köhler, B. Ludäscher, D. Zinn, First-Order Provenance Games, in: *In Search of Elegance in the Theory and Practice of Computation*, LNCS 8000, 2013, pp. 382–399.
- [10] P. T. Wood, Query Languages for Graph Databases, *ACM SIGMOD Record* 41 (2012) 50–60.
- [11] M. Caminada, P. Dunne, Strong Admissibility Revisited: Theory and Applications, *Argument & Computation* 10 (2020) 277–300.
- [12] T. Bench-Capon, Representation of Case Law as an Argumentation Framework, in: *JURIX Conf. on Legal Knowledge and Information Systems*, 2002, pp. 103–112.
- [13] Y. Xia, S. Bowers, B. Ludäscher, Demonstrating Provenance for Grounded Argumentation, 2024. <https://github.com/idaks/Games-and-Argumentation/tree/safa2024>.
- [14] Y. Xia, D. Odenkerken, S. Bowers, B. Ludäscher, Layered Visualization of Argumentation Frameworks, *COMMA Demo-Paper Session*, 2024. (*to appear*).
- [15] B. Ludäscher, S. Bowers, Y. Xia, Games, Queries, and Argumentation Frameworks: Towards a Family Reunion, in: *Workshop on Advances in Argumentation in AI (AI³)*, volume 3546, CEUR, 2023.
- [16] A. Van Gelder, The Alternating Fixpoint of Logic Programs with Negation, *Journal of Computer and System Sciences* 47 (1993) 185–221.
- [17] S. Bowers, Y. Xia, B. Ludäscher, On the Structure of Game Provenance and its Applications, in: *Intl. Workshop on the Theory and Practice of Provenance (TaPP)*, 2024. (*to appear*).
- [18] P. Baroni, D. Gabbay, M. Giacomin, L. v. d. Torre, *Handbook of Formal Argumentation*, London, England: College Publications, 2018.
- [19] P. Baroni, M. Giacomin, On Principle-Based Evaluation of Extension-Based Argumentation Semantics, *Artificial Intelligence* 171 (2007) 675–700.
- [20] M. Caminada, On the Issue of Reinstatement in Argumentation, in: *Logics in Artificial Intelligence*, LNAI 4160, 2006, pp. 111–123.
- [21] T. M. J. Fruchterman, E. M. Reingold, Graph Drawing by Force-Directed Placement, *Software Practice & Experience* 21 (1991) 1129–1164.
- [22] E. R. Gansner, E. Koutsofios, S. C. North, K.-P. Vo, A Technique for Drawing Directed Graphs, *IEEE Trans. Softw. Eng.* 19 (1993) 214–230.
- [23] D. Odenkerken, A. Borg, M. Berthold, Demonstrating PyArg 2.0, in: *Workshop on Advances in Argumentation in AI (AI³)*, volume 3546, CEUR, 2023.
- [24] A. Borg, F. Bex, A Basic Framework for Explanations in Argumentation, *IEEE Intelligent Systems* 36 (2021) 25–35.
- [25] A. J. García, C. I. Chesñevar, N. D. Rotstein, G. R. Simari, Formalizing Dialectical Explanation Support for Argument-Based Reasoning in Knowledge-Based Systems, *Expert Systems with Applications* 40 (2013) 3233–3247.
- [26] X. Fan, F. Toni, On Computing Explanations in Argumentation, in: *AAAI*, 2015.