# Gradual handling of contradiction in argumentation frameworks

**C. Cayrol**
IRIT – UPS 118 route de Narbonne
31062 Toulouse Cédex
ccayrol@irit.fr

**M.C. Lagasquie**
IRIT – UPS 118 route de Narbonne
31062 Toulouse Cédex
lagasq@irit.fr

## Abstract

Argumentation is based on the exchange and evaluation of interacting arguments. In this paper, we concentrate on the analysis of defeat interactions between arguments, within the argumentation framework of [3]. We propose different principles for a gradual evaluation of arguments, which takes into account the defeaters, the defeaters of defeaters, and so on. Following these principles, two formalizations are presented. Finally, existing approaches are restated in our framework.

**Keywords:** Argumentation, Interactions, Inconsistency, Nonmonotonic logics

## 1 Introduction

As shown by [3], argumentation frameworks provide a unifying and powerful tool for the study of many formal systems developed for commonsense reasoning, as well as for giving meaning to logic programs. Argumentation is based on the exchange and evaluation of interacting arguments. It can be applied, among others, in the legal domain, for collective decision support systems or for negotiation support.

Two kinds of evaluation can be distinguished :

- intrinsic evaluation : an argument is considered independently of its interactions with other arguments. This enables to express to what extent an argument increases the confidence in the statement it supports. Such valuation methods can take different forms [5], including a numerical value which can be interpreted in probabilistic terms as in [6], or simply by a preference relation on the set of all arguments as in [7] and [1].
- interaction-based evaluation : an argument is valuated according to its defeaters, to the defeaters of its defeaters (the defenders), ...

Intrinsic evaluation and interaction-based evaluation have often been used separately, according to the considered applications. Some recent works however consider a combination of both approaches (see eg. [1]). Interaction-based approaches usually provide a crisp evaluation : the argument is accepted or not. Such an evaluation can be defined individually (per argument), for example by a labelling process [4], or globally, giving sets of arguments that are all accepted together [3]. The common idea is that an argument is acceptable if it can be argued successfully against defeating arguments. Recently, [2] has proposed a gradual interaction-based evaluation in the specific case of deductive arguments (where an argument is a logical proof). An argument tree represents all the chains of defeaters for the argument at the root of the tree. The (computed) value of this tree represents the relative strength of the root argument.

In this paper, we concentrate on the analysis of interactions between arguments, abstracting from the internal structure of an argument. We therefore ignore intrinsic evaluations. Our purpose is to provide a global evaluation of an argument according to the way this argument is defeated by other arguments and by recursion on these defeaters. An argument will be thus considered as more or less acceptable.

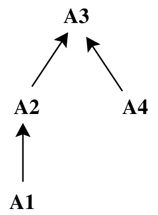
Section 2 introduces abstract argumentation frameworks and their representation as attack graphs. In section 3, we present relevant examples that enable us to draw the main principles underlying a gradual evaluation of interactions. Following these principles, two formalizations are proposed in section 4, and related to other existing approaches in section 5.

## 2 Notations

We consider the abstract framework introduced in [3]. An *argumentation system* $<\mathcal{A}, \mathcal{R}>$ is a set $\mathcal{A}$ of arguments and a binary relation $\mathcal{R}$ on $\mathcal{A}$ called an *attack relation*: let $A_i$ and $A_j \in \mathcal{A}$, $A_i \mathcal{R} A_j$ means that $A_i$ attacks $A_j$.

**Notations:** Let $A \in \mathcal{A}$, The set $\{A_i \in \mathcal{A} | A_i \mathcal{R} A\}$ is denoted by $\mathcal{R}^-(A)$ and the set $\{A_i \in \mathcal{A} | A \mathcal{R} A_i\}$ is denoted by $\mathcal{R}^+(A)$.

$<\mathcal{A}, \mathcal{R}>$ defines a directed graph $\mathcal{G}$ (called the *attack graph*). For example, the system $<\mathcal{A} = \{A_1, A_2, A_3, A_4\}, \mathcal{R} = \{(A_2, A_3), (A_4, A_3), (A_1, A_2)\}>$ defines the following graph $\mathcal{G}$:




**Def 1** *An argument $A \in \mathcal{A}$ such that $\mathcal{R}^-(A) = \varnothing$ is a* leaf of the attack graph *defined by $<\mathcal{A}, \mathcal{R}>$.*

**Def 2** *In the attack graph $\mathcal{G}$, a* path from $A$ to $B$*, denoted by $\mathcal{C}(A, B)$*[1]*, is a sequence of arguments $A_1 \dots A_n$ such that $A = A_1$, $A_1 \mathcal{R} A_2, \dots, A_{n-1} \mathcal{R} A_n$, $A_n = B$. The* length of this path *is $n-1$ (the number of edges that are used in the path) and will be denoted by $l_{\mathcal{C}(A,B)}$.*

We now introduce the notions of direct and indirect attacks and defences. The notions introduced here are inspired by related definitions first introduced in [3] but are not strictly equivalent: in [3]'s work, direct attacks (resp. defences) are also indirect attacks (resp. defences) which is not true in our definitions:

[1] If there are several paths from $A$ to $B$, they are denoted by $\mathcal{C}(A, B)_1, \mathcal{C}(A, B)_2 \dots$

**Def 3** *Let $A \in \mathcal{A}$. The* direct defeaters *of $A$ are the elements of $\mathcal{R}^-(A)$. The* direct defenders *of $A$ are the direct defeaters of the elements of $\mathcal{R}^-(A)$. The* indirect defeaters *of $A$ are the elements $A_i$ such that: $\exists \mathcal{C}(A_i, A)$ such that $l_{\mathcal{C}(A_i,A)} = 2k+1$ with $k \geq 1$. The* indirect defenders *of $A$ are the elements $A_i$ such that: $\exists \mathcal{C}(A_i, A)$ such that $l_{\mathcal{C}(A_i,A)} = 2k$ with $k \geq 2$.*

**Def 4** *Let $A \in \mathcal{A}$, an* attack branch *(resp.* defence branch*) for $A$ is an odd (resp. even) length path from a leaf to $A$ in $\mathcal{G}$. When such a branch exists, $A$ is said to be the root of the branch.*

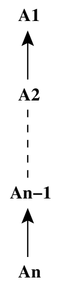
## 3 Examples

We have identified some specific relevant graphs which allow us to draw some fundamental principles and also problems that will lead to the introduction of two different formalizations of gradual evaluation.

We denote by $v$ an assignment of values from a given ordered set to the arguments from $\mathcal{A}$.

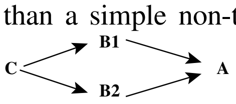
1. case of a single node $A$: The value of $A$ is maximal since it suffers no attack.

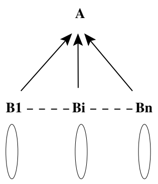
2. case of a single path of length $n \geq 1$:

   According to the parity of $n$, we have different results. If $n$ is even, the path $\mathcal{C}(A_n, A_1)$ is an attack branch for $A_1$ and the $A_i$ such that $i$ odd, $1 < i \leq n-1$, are defenders of $A_1$ and the other $A_i$ are defeaters of $A_1$ ; else (if $n$ odd) the path $\mathcal{C}(A_n, A_1)$ is a defence branch for $A_1$ and the $A_i$ such that $i$ odd, $1 < i \leq n$, are defenders of $A_1$ and the other $A_i$ are defeaters of $A_1$.

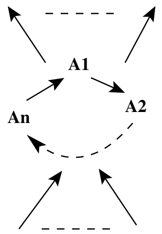
   So, we have either $v(A_n) \geq v(A_{n-2}) \geq \dots \geq v(A_2) \geq v(A_1) \geq v(A_3) \geq \dots \geq v(A_{n-3}) \geq v(A_{n-1})$ (for $n$ even), either $v(A_n) \geq v(A_{n-2}) \geq \dots \geq v(A_3) \geq v(A_1) \geq v(A_2) \geq \dots \geq v(A_{n-3}) \geq v(A_{n-1})$ (for $n$ odd).

3. case of a non-linear acyclic attack graph (note that this type of graph is more general than a simple non-trivial tree; for example 
   
   is a non-linear acyclic attack graph and is not a non-trivial tree):




This example raises the issue of the propagation of the modification of the value of a direct defeater if there are several defeaters.

4. case of an attack graph with cycles:




How to take into account the cycles (even or odd) and their lengths ?

## 4 Principles and formalization

We will consider two types of evaluation:

1. the evaluation of an argument is only a function of the direct defeaters of this argument. Therefore, defenders are taken into account through the defeaters.

2. the evaluation of an argument is a function of the set of all attack and defence branches for this argument.

### 4.1 Approach 1

***Underlying principles*** They are 4. **P1:** The evaluation is maximal for an unattacked argument and non maximal for an attacked and undefended argument. **P2:** The evaluation of an argument is a function of the evaluation of its direct defeaters (the "direct attack"). **P3:** The evaluation of an argument is a decreasing function of the evaluation of the direct attack. **P4:** Each defeater of an argument contributes to the increase of the evaluation of the direct attack for this argument.

***Formalization*** Let $W$ be a totally ordered set with a minimum element ($V_{\text{Min}}$) and a subset $V$ of $W$ that contains $V_{\text{Min}}$ and with a maximum element $V_{\text{Max}}$.

**Def 5** *Let $<\mathcal{A}, \mathcal{R}>$ be an argumentation system. An evaluation is a function $v : \mathcal{A} \to V$ such that:*

1. $\forall A \in \mathcal{A}, v(A) \geq V_{Min}$
2. $\forall A \in \mathcal{A}$, *if* $\mathcal{R}^-(A) = \varnothing$ *then* $v(A) = V_{Max}$
3. $\forall A \in \mathcal{A}$, *if* $\mathcal{R}^-(A) = \{A_1, \ldots, A_n\} \neq \varnothing$ *then* $v(A) = g(h(v(A_1), \ldots, v(A_n)))$

*where $h : V^* \to W$ is such that ($V^*$ denotes the set of all finite sequences of elements of $V$) $h(x) = x$, $h() = V_{Min}$, $h(x_1, \ldots, x_n, x_{n+1}) \geq h(x_1, \ldots, x_n)$*

*and $g : W \to V$ is such that $g(V_{Min}) = V_{Max}$, $g(V_{Max}) < V_{Max}$, $g$ is decreasing ($x \leq y \Rightarrow g(x) \geq g(y)$).*

Note: the fact that $h(x_1, \ldots, x_n) \geq max(x_1, \ldots, x_n)$ is a logical consequence of the properties of $h$.

***Properties*** **Existence of an evaluation** $v$: see section 5 (property 3) for an example of an evaluation introduced in the framework of [4].

**P1** is satisfied since $\forall A \in \mathcal{A}$ if $A$ has no defeater ($\mathcal{R}^-(A) = \varnothing$), then $v(A) = V_{\text{Max}}$ and $g(V_{\text{Max}}) < V_{\text{Max}}$. **P2** is satisfied since if $\mathcal{R}^-(A) = \{A_1, \ldots, A_n\}$, $h(v(A_1), \ldots, v(A_n))$ gives an evaluation of the direct attack of $A$. **P3** and **P4** are satisfied due to the properties of the functions $g$ and $h$.

**Prop 1** *The function $g$ satisfies for all $n \geq 1$:*

$$g(V_{Max}) \leq g^3(V_{Max}) \leq \ldots \leq g^{2n+1}(V_{Max})$$
$$\leq g^{2n}(V_{Max}) \leq \ldots \leq g^2(V_{Max}) \leq V_{Max}$$

*If, moreover, $g$ is strictly decreasing and $g(V_{Max}) > V_{Min}$, the inequalities become strict.*

**Proof:** By induction from $V_{\text{Min}} \leq g(V_{\text{Max}}) < V_{\text{Max}}$ and by applying $g$ twice consecutively. □

***Examples*** Let us consider again some examples from section 3.

**Ex 1** $v(A) = V_{\text{Max}}$

**Ex 2** $v(A) = g^{n-1}(V_{\text{Max}})$

If $n$ is even $v(A_{n-1}) \leq \ldots \leq v(A_3) \leq v(A_1) \leq v(A_2) \leq \ldots \leq v(A_n) = V_{\text{Max}}$

If $n$ is odd $v(A_{n-1}) \leq \ldots \leq v(A_2) \leq v(A_1) \leq v(A_3) \leq \ldots \leq v(A_n) = V_{\text{Max}}$

**Ex 3** From the properties of $g$ and $h$, we conclude that if $v(B_i)$ increases (resp.

decreases), then $v(A)$ decreases (resp. increases). A particular case is the addition of a new defeater $B_{n+1}$: $v(A) = g(h(v(B_1), \ldots, v(B_{n+1}))) \leq g(h(v(B_1), \ldots, v(B_n)))$. This example shows that if the number of defeaters is taken into account, then the evaluation of an argument should be able to be strictly lower than $g(V_{\text{Max}})$. This is for example the case if $h$ is strictly decreasing.

**Ex 4** The study of the graphs with cycles brings to light the role played by the fixpoints of $g$, $g^2, \ldots$. In the case of cycles of length $n$: if $n$ is odd, all the arguments of the cycle have the same value and this value is a fixpoint of $g$, if $n$ is even, the values of all the arguments in the cycle are fixpoints of $g^n$.

Table 1: Synthesis of the possible values

| $V_{\text{Min}}$ | | $g(V_{\text{Max}})$ | | $V_{\text{Max}}$ |
|---|---|---|---|---|
| An infinity of direct defeaters | A single level of attack. More than a direct defeater. | A single level and a single direct defeater. | Several levels and a single defeater per level. | Unattacked argument |
| | Several levels and more than one defeater per level | | | |

## 4.2 Approach 2

***Underlying principles*** There are some common items with the previous approach. **P1':** The evaluation is maximal for an argument without defeater and non maximal for an argument which is attacked (whether it is defended or not). **P2':** The evaluation of an argument takes into account all the branches which are rooted in this argument. **P3':** The improvement of the defence or the degradation of the attack of an argument leads to an increase of the value of this argument. **P4':** The improvement of the attack or the degradation of the defence of an argument leads to a decrease of the value of the argument. Let us define a *tuple-based evaluation*.

***Formalization*** Let us define recursively the value (a tuple) of an argument $A$. We have two cases:

**Rule 1 ($A$ does not belong to a cycle)** *Let $A$ be an argument which does not belong to a cycle, the value $v(A)$ of $A$ is defined as follows:*

- *if $A$ is a leaf of the attack graph: $v(A) = ()$ (also denoted by $v(A) = (0)$),*
- *else: let $B_1, \ldots, B_n$ be the direct defeaters of $A$ whose respective values are tuples of integers ordered by increasing values $(b_1^1, \ldots, b_{m_1}^1), \ldots, (b_1^n, \ldots, b_{m_n}^n)$, then $v(A) = f_c(b_1^1 + 1, \ldots, b_{m_1}^1 + 1, \ldots, b_1^n + 1, \ldots, b_{m_n}^n + 1)$, where $f_c$ is the function that orders a tuple by increasing values.*

The incrementation of each value in the tuple means that the argument $A$ is more distant of the leafs than its defeaters. Second, the function $f_c$ guarantees that each tuple is ordered.

When cycles occur, we have chosen to memorize only two "branches" (abuse of notation !): the attack branch of minimal length and the defence branch of minimal length. The underlying idea is to consider each cycle as a kind of meta-argument whose value is equal to the tuple $(1, 2)$. The values of the arguments inside the cycle will be defined as the tuple built from the value of the meta-argument and from the values of the arguments that attack the meta-argument *seen from the argument considered*. This leads to the following computation rule:

**Rule 2 ($A$ belongs to a cycle)** *Let $A$ be an argument, $A$ belongs to $\mathtt{C}_1 \ldots \mathtt{C}_m$ cycles. Let $\mathtt{C}'_1 \ldots \mathtt{C}'_n$ other cycles which are interconnected with an $\mathtt{C}_i$ cycle or between them*[2] *(so $A$ does not belong to $\mathtt{C}'_1 \ldots \mathtt{C}'_n$). Let $X^1 \ldots X^p$ arguments which do not belong to the cycles and which are direct defeaters of an element of a cycle. We denote :*

- *$l_i$ : the shortest distance between the cycle $\mathtt{C}'_i$ and $A$ ($l_i = Min_{C'_j \in \mathtt{C}'_i}(l_{\mathcal{C}(C'_j, A)})$),*
- *$l_{X^i} = l_{\mathcal{C}(X^i, A)}$ (distance between $X^i$ et $A$),*
- *each argument $X^i$ has the following value : $v(X^i) = (x_1^i, \ldots, x_{k^1}^i)$.*

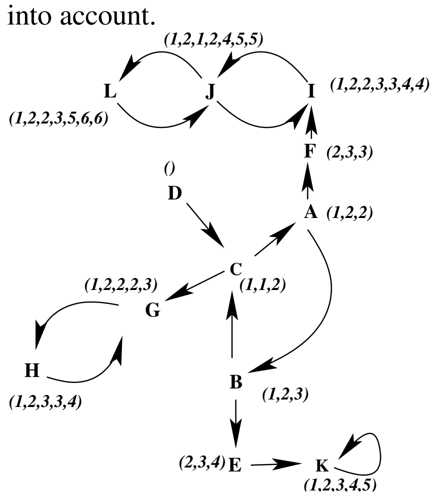
*So, the value $v(A)$ of $A$ is defined as follows :*

$$v(A) \quad = \quad f_c(\overbrace{1, 2, \ldots, 1, 2,}^{}_{m \text{ times}}$$

[2] 2 cycles are interconnected iff their intersection is no empty.

$$
\begin{aligned}
&1+l_1, 2+l_1, \ldots, 1+l_n, 2+l_n,\\
&x_1^1 + l_{X^1}, \ldots, x_{k^1}^1 + l_{X^1},\\
&\ldots,\\
&x_1^p + l_{X^p}, \ldots, x_{k^p}^p + l_{X_p})
\end{aligned}
$$

Here is a example that summarizes the various cases that can be encountered when cycles are taken into account.




We then consider an algorithm that compares two values (ordered tuples of integers). See algorithm 1 where the following notations are used:

Let $X$ be an argument and $v(X)$ its value. $n_a(X)$ is the number of attack branches of $X$ (0 if $X$ is a leaf, the number of odd integers in the tuple $v(X)$ otherwise). $n_d(X)$ is the number of defence branches of $X$ ($\infty$ if $X$ is a leaf, the number of even integers in the tuple $v(X)$ otherwise). $v_p(X)$ (resp. $v_i(X)$) is the ordered tuple of even (resp. odd) integers in the tuple $v(X)$.

**Justification** The underlying idea is that an argument $A$ will be better than an argument $B$ if it as a "better" defence" and a "weaker" attack.

In order to compare two arguments $A$ and $B$, we start from the numbers $n_a(A)$, $n_a(B)$ on one side and $n_d(A)$, $n_d(B)$ on the other side (for a quantitative aspect). We then aggregate these two criteria in the most cautious way (the two criteria have to be in agreement to conclude).

In case of equality of $n_a(A)$, $n_a(B)$ and of $n_d(A)$, $n_d(B)$, we have chosen to take into account the "quality" of the attack and defence branches (again we get two criteria for the comparison), represented by their length. We have chosen to compare each of these criteria lexicographically. Then we cautiously aggregate the results of these two comparisons, as above.

Note that, instead of a lexicographic comparison, one may rely on other mechanisms. Similarly, other types of aggregation may be used.

**Algorithm 1:** Comparison of two values (tuples)

% **Description of parameters:** %
% $v(A)$, $v(B)$: 2 tuples %

**begin**
- **if** $((n_a(A) > n_a(B)) \wedge (n_d(A) \leq n_d(B))) \vee ((n_a(A) \geq n_a(B) \wedge (n_d(A) < n_d(B)))$ **then**
  - $v(B) > v(A)$ % Case 1 %
- **else**
  - **if** $((n_a(A) < n_a(B)) \wedge (n_d(A) \geq n_d(B)) \vee ((n_a(A) \leq n_a(B)) \wedge (n_d(A) > n_d(B)))$ **then**
    - $v(A) > v(B)$ % Case 2 %
  - **else**
    - **if** $(n_a(A) = n_a(B)) \wedge (n_d(A) = n_d(B))$ **then**
      - % lexicographical comparisons % % of $v_p(A)$ and $v_p(B)$ and of % % $v_i(A)$ and $v_i(B)$ %
      - **if** $(v_p(A) \leq_{lex} v_p(B)) \wedge (v_i(A) \geq_{lex} v_i(B))$ **then**
        - $v(A) \geq v(B)$ % Case 3 %
      - **else**
        - **if** $(v_p(A) \geq_{lex} v_p(B)) \wedge (v_i(A) \leq_{lex} v_i(B))$ **then**
          - $v(B) > v(A)$ % Case 4 %
        - **else**
          - % Case 5 % $v(B) \not\geq v(A)$ and $v(A) \not\geq v(B)$
    - **else**
      - % Case 6 % $v(B) \not\geq v(A)$ and $v(A) \not\geq v(B)$

**end**

***Properties*** From the previous definitions, we get the following results:

**Prop 1** *The relation defined by the comparison algorithm 1 is a partial order on the set $V$ of all the finite tuples of integers ordered by increasing values and $()$ is the maximum element of $V$.*

**Proof:** The fact that the relation is an order relation (reflexive, antisymmetric and transitive) and that () is a maximum element of this relation follows from the definition of the comparison algorithm 1. □

We then define a *per tuple evaluation*:

**Def 6** *Let $V$ be the set of all the finite tuples of integers, ordered by increasing values. Let $<\mathcal{A}, \mathcal{R}>$ be an argumentation system. The binary relation $v$ on $\mathcal{A} \times V$ such that, for any $A \in \mathcal{A}$, $v(A) \in V$ and $v(A)$ satisfies the computation rules 1 and 2 is called a* per tuple evaluation *of $<\mathcal{A}, \mathcal{R}>$.*

**Prop 2** *Let $v$ be a per tuple evaluation of $<\mathcal{A}, \mathcal{R}>$, $v$ satisfies the principles* **P1'**, **P2'**, **P3'** *and* **P4'**.

**Proof: P1'** is satisfied due to the use of the computation rules 1 and 2 and from the fact that () is the maximum element of $V$.

**P2'** is satisfied due to the computation rules 1 and 2.

**P3'** and **P4'** are satisfied. It can be checked by examining, one after the other, all the basic possible cases of modification of a value and by comparing these values before and after the modification using the algorithm 1. For principle **P3'**, these different basic cases are removal or lengthening of an attack branch, addition or shortening of a defence branch. And for principle **P4'**, they are removal or lengthening of a defence branch, addition or shortening of an attack branch. Naturally, the change in the length of a branch should not modify the type of the branch itself or else we are no longer in a basic case but in the complex case of a branch removal from a given nature followed by the addition of a branch of another type. □

***Examples*** On the examples cited in section 3, we get the following results:

**Ex 1** $v(A) = ()$ (max of $V$).

**Ex 2** If $n$ is even, $v(A_n) = () > v(A_{n-2}) = (2) > \ldots > v(A_2) = (n-2) > v(A_1) = (n-1) > \ldots > v(A_{n-3}) = (3) > v(A_{n-1}) = (1)$. For $n$ odd, $v(A_n) = () > v(A_{n-2}) = (2) > \ldots > v(A_1) = (n-1) > v(A_2) = (n-2) > \ldots > v(A_{n-3}) = (3) > v(A_{n-1}) = (1)$.

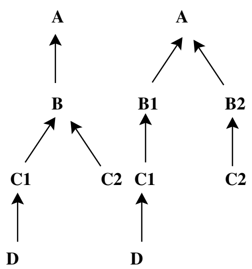
**Ex 3** If $B_i$ is a defence branch for $A$ and if $B_i$ disappears, then $v(A)$ will decrease. If $B_i$ is an attack branch for $A$ whose length increases then $v(A)$ will increase (see principles **P3'** and **P4'**). In the case of the addition of a new defeater $B_{n+1}$, if $B_{n+1}$ is on a defence branch for $A$ then $v(A)$ will increase, and if $B_{n+1}$ is on an attack branch for $A$, $v(A)$ will decrease (see principles **P3'** and **P4'**).

**Ex 4** Generally, the value of the arguments in the cycle is $(1, 2) < ()$.

### 4.3 Comparison of these two approaches

Even if these two approaches rely on different underlying principles, they may yield the same results in some cases. This is the case for example when the attack graph reduces to a unique branch (see example 2). Differences appear as soon as, for a given argument, it is possible to get both attack and defence branches (see examples 3 and the following one).

This is due to the fact that the approach 1 aggregates the values of the direct defeaters before evaluating the argument itself whereas approach 2 does not aggregate any value and treats each branch independently.




Considering cycles, the approach 1 necessitates the existence of fixpoints for the evaluation functions which is absolutely not needed for the approach 2.

## 5 Comparison with existing proposals

### 5.1 The approach of [3]

The argumentation schema introduced in [3] enables the definition of sets of acceptable arguments, called extensions, along with different possible semantics. Under the preferred extension semantics, an extension is an inclusion-maximal conflict-free set which defends all its elements. An argument is therefore accepted iff it belongs to a preferred extension. One may therefore consider that for each semantics, there is a corresponding type of binary interaction-based evaluation and, under a given semantics, a binary evaluation corresponds to each extension.

### 5.2 The approach of [4]

In the approach of [4], a labelling of a set of arguments assigns a status (accepted, rejected, undecided) to each argument using labels from the set $\{+, -, ?\}$. $+$ (resp. $-$, ?) represents the "accepted" (resp. "rejected", "undecided") status. Intuitively, an argument labelled with ? is both supported and weakened.

**Def 7** *Let $<\mathcal{A}, \mathcal{R}>$ be an argumentation system. A complete labelling of $<\mathcal{A}, \mathcal{R}>$ is a function $Lab : \mathcal{A} \to \{+, ?, -\}$ such that:*

1. *If $Lab(A) \in \{?, -\}$ then $\exists B \in \mathcal{R}^-(A)$ such that $Lab(B) \in \{+, ?\}$*
2. *If $Lab(A) \in \{+, ?\}$ then $\forall B \in \mathcal{R}^-(A) \cup \mathcal{R}^+(A)$, $Lab(B) \in \{?, -\}$*

The underlying intuition is that an argument can only be weakened (label $-$ or ?) if one of its direct defeaters is supported (condition 1); an argument can get a support only if all its direct defeaters are weakened and an argument which is supported (label $+$ or ?) weakens the arguments it attacks (condition 2). So:

- If $A$ has no defeater $Lab(A) = +$.
- If $Lab(A) = ?$ then $\exists B \in \mathcal{R}^-(A)$ such that $Lab(B) = ?$.
- If $(\forall B \in \mathcal{R}^-(A),\ Lab(B) = -)$ then $Lab(A) = +$.
- If $Lab(A) = +$ then $\forall B \in \mathcal{R}^-(A) \cup \mathcal{R}^+(A)$, $Lab(B) = -$.

Every argumentation system can be completely labelled. The associated semantics is that $S$ is an acceptable set of arguments iff there exists a complete labelling $Lab$ of $<\mathcal{A}, \mathcal{R}>$ such that $S = \{A | Lab(A) = +\}$.

Other types of labellings are introduced in [4] among which the so-called "rooted labelling" which induces a corresponding "rooted" semantics. The idea is to only reject arguments attacked by accepted arguments: an attack by an "undecided" argument is not rooted since an "undecided" defeater may become rejected.

**Def 8** *The complete labelling $Lab$ is rooted iff $\forall A \in \mathcal{A}$, if $Lab(A) = -$ then $\exists B \in \mathcal{R}^-(A)$ such that $Lab(B) = +$.*

The rooted semantics allows to clarify the links between all the other semantics introduced in [4] and some semantics introduced in [3].

***On the examples of section 3*** We get the following results:

**Ex 1** $v(A) = +$

**Ex 2** For $n$ even, $v(A_n) = v(A_{n-2}) = \ldots = v(A_2) = +$ and $v(A_{n-1}) = v(A_{n-3}) = \ldots = v(A_1) = -$. For $n$ odd, $v(A_n) = v(A_{n-2}) = \ldots = v(A_1) = +$ and $v(A_{n-1}) = v(A_{n-3}) = \ldots = v(A_2) = -$

More generally, we proved:

- In the directed acyclic graph case, every argument is labelled by $+$ or $-$. An argument $A$ is accepted iff it is (indirectly) defended against each of its direct defeaters (i.e., iff each of its direct defeaters is rejected).
- In the case of a graph with cycle: the arguments that appear on an odd length cycle are necessarily labelled by "undecided". If $Lab$ is a rooted labelling:
  - $A$ is accepted ($+$) iff all its direct defeaters are rejected ($-$).
  - $A$ is rejected iff one of its direct defeaters is accepted.
  - $A$ is undecided iff one of its direct defeaters is undecided and none of its direct defeaters is accepted.

**Prop 3 (Link with the approach 1)** *Every rooted labelling of $<\mathcal{A}, \mathcal{R}>$ can be defined by an evaluation $v$ such that: $V = W = \{-, ?, +\}$ with $- < ? < +$, $V_{Min}$ denoted by $-$ and $V_{Max}$ denoted by $+$.*

> **Proof:** $g$ is defined by $g(-) = +$, $g(+) = -$, $g(?) = ?$ and $h$ is the function $\max$. $\square$

### 5.3 The aggregation function of [2]

[2] introduces a logical framework for "deductive" arguments. The approach can be characterized as follows. An argument is structured as

a pair $\langle support, conclusion \rangle$, where *support* is a consistent set of formulae that enables to prove the formula *conclusion*. The attack relation considered here is strict and cycles are not allowed. The notion of a "tree of arguments" allows a concise and exhaustive representation of defeaters and defenders of a given argument, root of the tree. A function, called a "categorizer", allows to assign a value to a tree of arguments. This value represents the relative strength of an argument (root of the tree) given all its defeaters and defenders. A function, called an "accumulator", allows to synthesize the values assigned to all the argument trees whose root is an argument for (resp. against) a given conclusion.

The phase of categorization therefore corresponds to an interaction-based evaluation. [2] introduces the following function $Cat$:

- if $\mathcal{R}^-(A) = \varnothing$, then $Cat(A) = 1$
- if $\mathcal{R}^-(A) \neq \varnothing$ with $\mathcal{R}^-(A) = \{A_1, \ldots, A_n\}$, $Cat(A) = \frac{1}{1+Cat(A_1)+\ldots+Cat(A_n)}$

Intuitively, the larger the number of direct defeaters of an argument, the lower its value. The larger the number of defenders of an argument, the larger its value.

***On the examples of section 3*** We get the following results:

**Ex 1** $v(A) = 1$

**Ex 2** We get the same results as in section 4.1

**Prop 4 (Link with the approach 1)** *The categorization function of [2] is the restriction to acyclic graphs of an evaluation function as introduced in section 4.1.*

> **Proof:** Let us consider $V = [0, 1]$, $W = [0, \infty[$, $V_{\mathrm{Min}} = 0$ and $V_{\mathrm{Max}} = 1$. We define $g : W \to V$ as $g(x) = \frac{1}{1+x}$ and $h$ by $h(\{x_1, \ldots, x_n\}) = x_1 + \ldots + x_n$. □

The evaluation used in the above proof can be used for any attack graph (acyclic or not). The function $g$ defined has a single fixpoint $\alpha = (\sqrt{5} - 1)/2$. This is the inverse of the golden ratio. This is also the unique fixpoint of $g^2$. Consequently, every argument in a cycle will be assigned this value.

## 6 Conclusion

This paper gives a first analysis of gradual handling of defeat interaction between arguments. For that purpose, we have chosen the abstract argumentation framework of Dung, represented by an attack graph. Considering some relevant examples, we have drawn principles for a gradual evaluation, and proposed two different formal models. One of these models generalizes a previous proposal for handling interaction between deductive arguments. An extension of this work is to take into account intrinsic values of the arguments in order to define combined evaluation models.

### Acknowledgements

Thanks to Thomas SCHIEX and to Philippe BESNARD for their help.

## References

[1] Leila Amgoud and Claudette Cayrol. On the acceptability of arguments in preference-based argumentation. In G. F. Cooper and S. Moral, editors, *Proc. of the 14$^{th}$ Uncertainty in Artificial Intelligence*, pages 1–7, Madison, Wisconsin, 1998. Morgan-Kaufmann.

[2] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128 (1-2):203–235, 2001.

[3] Phan Minh 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.

[4] H. Jakobovits and D. Vermeir. Robust semantics for argumentation frameworks. *Journal of logic and computation*, 9(2):215–261, 1999.

[5] P. Krause, S. Ambler, M. Elvang, and J. Fox. A logic of argumentation for reasoning under uncertainty. *Computational Intelligence*, 11 (1):113–131, 1995.

[6] S. Parsons. Normative argumentation and qualitative probability. *LNAI*, 1244:466–480, 1997.

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