Argumentation among Agents: Review and Commentary

Grigore Costin-Teodor Radu Ștefan-Octavian
Vasiliu Florin Vintilă Eduard

1 Introduction

The present essay is, essentially, a summary of Iyad Rahwan's *Argumentation among Agents*, which appears as Chapter 5 in [10]. Where feasible, we have tried to phrase the given definitions more clearly, as well as to provide our own examples. The material is covered in the same order as in the given reference.

2 Several notions of "argumentation"

Before passing on to the "multiagent" half of the title, the author exhibits three formalizations of the "argumentation" concept, all pre-existent in the literature. By employing mathematical descriptions, these differ from the philosopher's viewpoint, which is expressed informally: an argument consists of the giving of claims in favor or against a statement that is open for debate.

2.1 Prakken's framework

Since this first framework is stated for reference, but used infrequently, we take the liberty of presenting a trimmed down version, that nevertheless contains all the components which the book elaborates upon.

As can be seen more clearly in Prakken's original article [8], his "argumentation system" is very similar to common logics—the major difference being the existence of two distinct kinds of inference rules: *strict* and *defeasible*. (Note that "defeasible" means "open to objection".)

Concretely, we take an *argumentation system* to be a tuple $(\mathcal{L}, cont, S, D)$, where $\mathcal{L}$ is a "logical language" (not much more is given about it, except that it should contain negation), *cont* is the so-called *contrariness* function, and

S, D are, respectively, the strict and defeasible sets of inference rules. With respect to the *cont* function, we note that it takes a member of $\mathcal{L}$ (henceforth referred to as a *formula*), and returns a set of formulas. Its purpose is to generalize ordinary negation. To show how, we extract again from Prakken [8, page 97]: if $\varphi \in cont(\psi)$, then

- If $\psi \notin cont(\varphi)$, then φ is a *contrary* of ψ;
- If $\psi \in cont(\varphi)$, then φ and ψ are *contradictory.*

Moreover, it is assumed that $\neg\varphi \in cont(\varphi)$ and $\varphi \in cont(\neg\varphi)$. Thus, a formula and its negation are contradictory, but neither is merely a contrary of the other.

We conclude this section by hinting at the definition of an *argument.* Note that inference rules transform a set of premises into a single conclusion (all of these are formulas). In writing, the distinction between strict and defeasible rules is given by the shape of the arrow symbol that is used. For instance, $p, q \rightarrow r$ indicates strictness: r surely follows from p and q. Defeasibility is illustrated thus: $u \Rightarrow v$, which informally means "v presumably follows from u". Starting from a set of formulas known as the *knowledge base*, an argument is then defined inductively to be one of the following:

- A member of the knowledge base.
- A tuple consisting of some arguments and a "conclusion" formula, written as $A_1, \ldots, A_n \rightarrow \varphi$, such that φ follows via a strict rule from the conclusions of $A_1, \ldots, A_n$.
- Likewise for a defeasible rule.

The complete framework also contains a partial order on the defeasible rules, indicating their "preferability". Using it, arguments may be compared. (In passing we note that Prakken's paper uses the baffling term "partial pre-order".)

2.2 Dung's model

We now skip directly to the argumentation model that is to be used most frequently in upcoming sections. Observe that Prakken's framework deals with the internal structure of arguments—their construction. The current model, due to Dung, considers arguments as completed entities. Following him, from now on, unless stated otherwise, an *argumentation framework* will be a finite directed graph, whose nodes are called "arguments". The

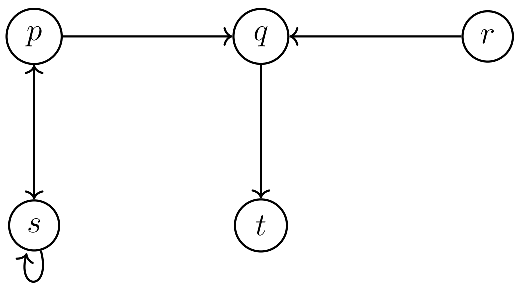


Figure 1: The running example.

adjacency relation is pronounced "defeats". Note that, as usual, this directed graph is a pair $(\mathcal{A}, \rightharpoonup)$, where $\mathcal{A}$ is a finite set and $\rightharpoonup$ is a binary relation on $\mathcal{A}$. Hence, $p \rightharpoonup q$ means "p defeats q". (Notationally, we have deviated from the author only in using Latin letters for variables ranging over arguments.) Henceforth, the discussion is conducted with respect to a fixed argumentation framework.

Our next task is to examine what makes an argument acceptable. In stating the following definitions, we adopt a primarily non-symbolic style of exposition, which facilitates understanding of the motivation behind them. As a further contribution, for illustration, we use our own example of an argumentation framework, visible in Figure 1. We shall refer to it as "the running example".

Definition 2.2.1 (+ and − operators) *For a set S of arguments, S^+ is the set of all arguments defeated by some member of S. For argument p, p^- is the set of all arguments which defeat p.*

For instance, in the running example, $\{p, q\}^+ = \{q, s, t\}$ and $s^- = \{p, s\}$.

Definition 2.2.2 (conflict-freeness, defending) *A set S of arguments is* conflict-free *if no argument in S defeats another also in S. Graph-wise, this means that there is no edge between two nodes in S. Also, we say S* defends *p if every argument which defeats p is defeated by S (i.e., is in S^+).*

In the running example, the sets $\{p, t\}$ and $\{r, t\}$ are conflict-free. It may also be said that $\{p, t\}$ defends t, since t is only defeated by q, and p defeats q.

Definition 2.2.3 (characteristic function) *The* characteristic function*, labelled $\mathcal{F}$, takes a set S of arguments and returns the set of arguments defended by S.*

In the running example, $\mathcal{F}(\{p, q, r\}) = \{r, t\}$ and $\mathcal{F}(\{r, t\}) = \{r, t\}$.

From this point on, we believe brevity surpasses transparency as a concern, so we revert to more symbolism.

Definition 2.2.4 (admissibility, complete extension) *Let S be a conflict-free set. S is* admissible *if $S \subseteq \mathcal{F}(S)$. (This latter condition means that S defends all its members.) If $S = \mathcal{F}(S)$ (i.e., S defends its members and nothing else), then S is a* complete extension.

By the previous remarks, it may be seen that $\{r, t\}$ is a complete extension.

Finally, we exhibit what the author calls "an equivalent characterization of complete extensions" (we shall supply a proof for one side of this equivalence—such a proof is not present in the chapter).

Definition 2.2.5 (labelling) *A labelling is a function L from $\mathcal{A}$ to $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$. It satisfies the following properties:*

- $L(p) = \texttt{in}$ *iff all attackers of p are* `out` *(under L),*
- $L(p) = \texttt{out}$ *iff some attacker of p is* `in`*,*
- $L(p) = \texttt{undec}$ *otherwise.*

We have used the expression "attacker of p" to designate a node which defeats p. The term seems more convenient in speech, although less proper than the author's "defeater".

It is claimed that, given a valid labelling, the nodes marked "`in`" constitute a complete extension.

Proof of claim Let S be the set of nodes labelled "`in`". First, we show that S is conflict-free. This can be seen immediately by the first condition imposed on a labelling: all attackers of members of S are `out`, and hence not in S.

Secondly, we prove the two necessary inclusions:

- $S \subseteq \mathcal{F}(S)$. Let $p \in S$. We have to show that p is defended by S. Let q be an attacker of p. By the properties of a labelling, q is `out`, and so it has an attacker that is `in`, i.e., $\in S$.
- $\mathcal{F}(S) \subseteq S$. Assume $p \notin S$. If p is `out`, then it has an attacker in S, which certainly cannot be defeated by S, since S is conflict-free. Hence, $p \notin \mathcal{F}(S)$. If p is `undec`, then it has an attacker q that is `undec` also. This q cannot be attacked by a member of S since that would make it `out`. Thus again $p \notin \mathcal{F}(S)$.

Indeed, for the running example, it may be checked that a function which assigns `in` to r, t, `out` to q, and `undec` to p, s, is a labelling.

Certain other kinds of extension (besides complete) are discussed. We do not delve into their details here. What is of interest is the next, and final, definition. It introduces the degrees of acceptability of an argument—their statement being the objective of this section.

Definition 2.2.6 *An argument p is:*

- skeptically accepted *iff p belongs to every extension,*
- credulously accepted *iff p belongs to some extension,*
- rejected *iff p doesn't belong to any extension.*

3 Argumentation Protocols

In the previous section, we have presented two possible models for formalizing the idea of argumentation: Prakken's framework and Dung's model. We find that both of them manage to capture the intuition behind the essence of an argument reasonably well. However, we shall see that the author focuses on the latter model to present a mechanism by which two agents can participate in a *dispute* where they can state and attack each other's arguments, much as in a real world debate.

As such, this section focuses on formalizing these "argumentation games" and also a set of constraints used in regulating the whole argumentation process in order to prevent situations where an agent might contradict himself, for example. These rules are called argumentation protocols and Dung's model proves to be particularly effective for them, since we do not have to deal with the internal structure of the arguments.

3.1 Argumentation Games

We begin by fixing the environment in which the argumentation process takes place. Essentially, we want to model a game where one player proposes an initial argument, after which both players take turns in counter-attacking the counterpart's last argument by specifying one which defeats it. The game shall be considered to be won by the player who lastly put forward an argument which could not be defeated by its opponent.

The author refers to Modgil and Caminada's work [6] for describing such a game, where the two players are defined as PRO (as in, the proponent which states the initial argument) and OPP (the opponent, which begins by counter-attacking the argument proposed by PRO). We provide the following formal definition of what represents a "dispute", a notion which captures the moves done by the players to attack the counterpart's arguments and is merely stated in natural language by the author:

Definition 3.1.1 (dispute) *Given an argumentation framework $(\mathcal{A}, \rightharpoonup)$, a dispute is a sequence $(a_k)_{k\in\mathbb{N}}$ (possibly infinite) of arguments from $\mathcal{A}$ where the following property holds: $\forall i \in \mathbb{N}^*, a_i \rightharpoonup a_{i-1}$ (i.e. every argument in the sequence defeats its preceding argument).*

From the above definition, we can thus consider that even indexed elements in the sequence (starting from a_0) are the arguments stated by the PRO player, and the odd ones by the OPP player. We can additionally define what it means to win a dispute:

Definition 3.1.2 (winning of a dispute) *A finite dispute $d = (a_k)_{k=0}^{n}$, where $n \in \mathbb{N}$ is the index of the last argument in d, is said to be won by the PRO player iff n is even and no argument from $\mathcal{A}$ can defeat a_n. Consequently, the dispute d is considered to be won by the OPP player iff n is odd and no argument from $\mathcal{A}$ can defeat a_n .*

This corresponds to the the established semantics by which a player wins only if its counterpart player cannot defeat the last argument that has been stated. Notice how the sequence of arguments must be finite in order to establish a winner - if the sequence would have been infinite, hence unbounded, we could not pronounce a winner since there would not exist a last argument in the dispute.

We observe that a player could potentially counter-attack its counterpart player with any argument whatsoever that defeats the last move. This leads to multiple disputes based on the defeating argument chosen by the player, which can be conveniently modeled as a dispute tree, as shown by the author. Such a tree can be constructed with the initial argument stated by the PRO player as a root node and child nodes as arguments which defeat the parent node. As such, all paths from root to leaf will represent all possible disputes for a given starting argument. We propose the following definition for a dispute tree, inspired by [6]:

Definition 3.1.3 (dispute trees) *Given an argumentation framework $(\mathcal{A}, \rightharpoonup)$ and an argument p in $\mathcal{A}$, a dispute tree induced by p is a tree T rooted in p, where each node represents an argument from $\mathcal{A}$ and for all arguments x, y in $\mathcal{A}$, x is a child of y in T iff x defeats y.*

Notice how each level of a dispute tree corresponds to arguments stated by either the PRO or the OPP player. If we count the levels of the tree starting from 0, then each even-numbered level marks arguments moved by the PRO player, and consequently odd-numbered levels represent arguments from the OPP player. As such, we can now consider a *dispute* merely a path in such a tree, and we shall see that further definitions and theorems stated by the author use this assumption.

However, since we have seen that disputes can potentially represent infinite sequences of arguments, this can lead, of course, to never-ending paths in a dispute tree where we cannot therefore pronounce a winner (we consider that a tree is finite if all of its paths are of finite length). Fortunately, we can avoid such situations by imposing a set of constraints by which a player cannot simply choose any argument that defeats its opponent's. One possible rule could describe that a player cannot state an argument which defeats any of its previous arguments in the dispute, for example (in other words, the set of arguments uttered by the player is *conflict-free*). By imposing such rules, we can capture various semantics of an argumentation game, such as a player cannot contradict himself, as is the case with the example rule that we have given. This is exactly the intuition behind an argumentation protocol, a notion which the author uses to introduce constraints that no longer allow infinite decision trees, through the definition of a protocol called G.

Definition 3.1.4 (dispute tree under protocol G) *Given a dispute tree T, we consider T to be under protocol G iff for any arbitrary dispute d in T and for any pair of arguments x, y stated by PRO in d, x is different than y.*

A dispute tree T under a protocol G essentially ensures that PRO does not state an argument which has been previously uttered by him. It is claimed by the author that the following holds:

Claim 1 *If T is a dispute tree under protocol G, then T is finite.*

Proof of claim Let T be a dispute tree under protocol G, induced by an argument p from an argumentation framework $(\mathcal{A}, \rightharpoonup)$, where $\mathcal{A}$ is a finite set of arguments with $n \in \mathbb{N}^*$ such that $card(\mathcal{A}) = n$. Let $d = (a_k)_{k \in \mathbb{N}}$ be a dispute in T of length of at least $2n$ arguments (we do not consider the other disputes, since we know they are of finite length).

We shall prove that d is of finite length; more specifically, exactly of length $2n$. We consider the argument a_{2n-1} from our sequence d. Without loss of generality, we establish $PRO_{2n-1}(d) = (a_{2k})_{k=0}^{n-1}$ to be the arguments uttered by PRO until argument a_{2n-1} (stated by OPP) from dispute d. By definition 3.1.4, we have that each argument in $PRO_{2n-1}(d)$ is pairwise different with all other arguments in it, hence all elements in its sequence are unique. Since $PRO_{2n-1}(d)$ is of length $n = card(\mathcal{A})$, every argument from $\mathcal{A}$ is found in this sequence. Now, assume that there exists an argument a_{2n} in dispute d. Since we have shown that all previous arguments uttered by PRO cover all the elements in $\mathcal{A}$, a_{2n} therefore must be identical to an argument already stated beforehand. This obviously contradicts the definition 3.1.4 for a dispute tree under the protocol G. As such, we have shown that a_{2n} doesn't exist, hence the sequence d ends at the a_{2n-1} argument, which means that d is of finite length $2n$.

Consequently, since no path (dispute) in T is of infinite length, this leads us to the conclusion that, indeed, T is finite.

4 Strategic Argumentation & Game Theory

What we've seen until now is an array of different argumentation protocols, which describe how an agent can move in different situations, given a set of possible moves. Variations among protocols can be observed. Some are "extremely prescriptive", only allowing agents to make a move at a time, while others such as the *persuasion protocol* proposed by Prakken allows agents multiple choices. The behaviour of agents is called a *strategy* and it influences both the outcome (e.g. the winner) and the dynamics (e.g. the pace) of the dialogue.

Researchers started exploring different types of strategies that agents could use in order to chose their arguments. Some of them came up with heuristic systems based on which agents evaluate or assert arguments.

Parsons et al. [7] developed a dialogue system, part of which they defined so called *agent attitudes*, such as *confident* (asserts all propositions for which he can construct an argument), *careful* (same as confident, but with the further restriction of not finding a stronger counterargument than the one found), *thoughtful* (asserts only propositions for which he finds acceptable arguments), etc. Other researchers have though of using social constructs (rights and obligations) [3], or the mental state of agents (beliefs, desires, intentions) [4].

The problem of the heuristic approach to designing argumentation strategies is that they only provide a subset of all possible strategies. However, it's been determined that an appropriate framework for doing a comprehensive study of **strategic argumentation** is provided by *game theory* (von Neuman and Morgenstern [9]).

Game theory can be used in the case of strategic argumentation in two main ways:

1. a thorough analysis of the strategic scenario can be tackled, focusing on predicting the outcome of the game; an agent can use this analysis to choose the best strategy for his exact scenario

2. design rules for the game (here, *argumentation protocol*) in such a way that the behaviour of the agents follows a desired path (e.g. promoting "good" argumentative behaviour); this is called *mechanism design* and will be further described in a following section.

4.1 Glazer and Rubenstein's Model

Glazer and Rubenstein were among the first to use *game theory* in the analysis of argumentation [2]. They used mechanism design in an attempt to create rules of debate such that a listener can arrive to the right conclusion when listening to arguments from two debaters. They described a minimal world state depicted by a vector $\omega = (w_1, w_2, \ldots, w_5)$ where each w_i is called an "aspect" and has values in the set $\{1, 2\}$. If $w_i = 1$ then w_i supports outcome O_1. If $w_i = 2$, then it supports outcome O_2. They model this setup as an extensive game, then consider different combinations of *procedural rules* (which determine the order and type of arguments each debater is allowed to state) and *persuasion rules* (which determine how the listener determines the outcome). Since the *persuasion rules* are chosen arbitrarily, there cannot be determined a rigorous correlation between the meaning of the arguments and the conclusion of the judge.

4.2 Game Theory Background

In this subsection core concepts of game theory are introduced in support of the following sections.

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i. Θ is the set of agent types; a type stands for private information of the agent and his preferences
- $o \in \mathcal{O}$ is an outcome; $\mathcal{O}$ is the set of all outcomes; an agent's preferences are related to outcomes
- $u_i(o, \theta_i)$ is a utility function which defines an agent's outcome preferences; if $u_i(o_1, \theta_i) > u_i(o_2, \theta_i)$, then agent i prefers outcome o_1 over outcome o_2
- Σ_i is the set of all strategies possible for an agent i and $s_i(\theta_i)$ is a strategy for agent i. We denote by *strategy* a detailed guide of what action to make in each possible situation.
- $s = (s_1(\theta_1), \ldots, s_I(\theta_I))$ is called a *strategy profile* and corresponds to the outcome of a setting where each agent i plays strategy $s_i(\theta_i)$
- The following are defined for convenience:

  $s_{-i}(\theta_{-i}) = (s_1(\theta_1), \ldots, s_{i-1}(\theta_{i-1}), s_{i+1}(\theta_{i+1}), \ldots, s_I(\theta_I))$

  $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_I)$

To clarify a bit, we observe that the *strategy profile* s has a shorter definition: $s = (s_i, s_{-i})$. Since s is an outcome ($s \in \mathcal{O}$), $u_i((s_i, s_{-i})), \theta_i)$ can be interpreted as the utility of agent i considering that all other agents use strategies denoted by s.

The fact that agents are **self-interested** is further emphasized. They will always chose strategies in order to maximize their own utility, but the choice is dependant on the strategies of the other agents. *Solution concepts* are thus introduced.

Definition 4.2.1 (Nash equilibrium [10]) *Let* $s^* = (s_1^*, \ldots, s_I^*)$ *be a* strategic profile*. Formally,* s^* *is a* Nash equilibrium *if the following holds:*

$$\forall i, \forall s_i^{'} \in \Sigma_i, u_i((s_i^*, s_{-i}^*), \theta_i) \geq u_i((s_i^{'}, s_{-i}^*), \theta_i).$$

Informally, s^* *is a* Nash equilibrium *if no agent would be better off utility-wise by changing its strategy, given that neither of the other agents changes its strategy.*

Although fundamental, the *Nash equilibrium* has various weaknesses. In a given scenario there could be multiple Nash equilibria, so agents don't have a deterministic way of choosing one. Moreover, to find a Nash equilibrium it is implicitly assumed that agents have perfect knowledge of all other agents's preferences.

The stronger *solution concept* of *dominant-strategy equilibrium* is thus introduced.

Definition 4.2.2 (Dominant Strategy [10]) *Formally, a strategy* $s_i^* \in \Sigma_i$ *is said to be* dominant *if*

$$\forall s_{-i}, \forall s_i^{'}, u_i((s_i^*, s_{-i}), \theta_i) \geq u_i((s_i^{'}, s_{-i}), \theta_i).$$

Informally, s_i^* *is* dominant *if agent* i *maximizes its utility, regardless of the strategies of the other agents.*

Following the above definition, a *dominant-strategy equilibrium* is a strategy profile where all strategies played by the agents are dominant. Compared to the *Nash equilibrium*, it is more solid as no information about other agents needs to be assumed. The downside is that there will be numerous settings where a dominant strategy cannot be found even for one agent.

4.2.1 Mechanism Design

The problem studied by mechanism design is that of achieving a desired outcome when dealing with a group of self-interested agents. The outcome usually depends on the preferences of the agents and can be described through what is called a *social choice* function.

Definition 4.2.3 (Social choice function) *A social choice function is defined as $f : \Theta_1 \times \cdots \times \Theta_I \to \mathcal{O}$, s.t $f(\theta) \in \mathcal{O}$ and $\theta = (\theta_1, \ldots, \theta_I)$. Informally, a social choice function matches agent types to outcomes.*

The *social choice function* is based on the agent type. As it is private information of the agent and the agent cannot be trusted to truthfully reveal his type (as it might be to his advantage to lie), a *mechanism* is used to reach the right outcome.

$\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is defined as a restricted set of strategies that the agents can choose from, agent i choosing from the set Σ_i. Also, let g be the function defined as $g : \Sigma \to \mathcal{O}$ and $g(s)$ the outcome o of the strategy $s = (s_1, \ldots, s_I) \in \Sigma$. A mechanism $\mathcal{M} = (\Sigma, g(\cdot))$ based on the above, defines a game where the choices of the agents are limited to Σ. It is said that the mechanism *implements* a social choice function f if the outcome induced by the mechanism is the same as the outcome returned by the function applied on the true types of the agents.

Definition 4.2.4 (Implementation [10]) $\mathcal{M} = (\Sigma, g(\cdot))$ *is a mechanism that implements the social function f, if there exists s^* an equilibrium s.t. $\forall \theta \in \Theta$, $g(s^*(\theta)) = f(\theta)$.*

Definition 4.2.5 (Direct-revelation mechanism [10]) *Formally, a mechanism is direct-revealing if $\forall i, \Sigma_i = \Theta_i$, and $\forall \theta \in \Theta, g(\theta) = f(\theta)$. Informally, the strategies of all agents are to announce a type $\theta_i^{'}$ to the* mechanism.

The above are a special class of *mechanism* where each agent's strategy is to announce a type $\theta_i^{'}$. It is said that a social function $f(\cdot)$ is *incentive compatible* if it can be *implemented* by a direct mechanism $\mathcal{M}$ where all agents reveal their true type. [5].

4.2.2 The Revelation Principle

Since the strategy search spaces are unlimited in the classical definition of a *mechanism*, finding a *mechanism* that implements a social function is hard. The *revelation principle* is thus useful, stating that the search can be limited only to a particular class of mechanisms [5].

Definition 4.2.6 (Revelation Principle [10]) *If considering dominant strategies, there exists a mechanism that implements social function f, then there exists a truthful direct mechanism that implements social function f, considering dominant strategies.*

The *revelation principle* states that the search field can be restricted to the class of truthful, direct mechanisms. We can conclude that no mechanism that implements the social function exists if we cannot find a mechanism in this restricted space.

5 The Argument Interchange Format

In the previous sections, we have elaborated on the semantics of arguments, the argumentation protocols, as well as the application of game theory in strategic argumentation. However, there has been no mention of a universal way of sharing the data underlying the arguments. This problem is especially evident in the practical implementations of such argumentation systems. Therefore, a community-led effort has laid down the foundations of a common, standardized specification for representing arguments and sharing integral data to their unfolding, called the *Argumentation Interchange Format (AIF)* [1].

5.1 Motivation for the AIF

The author of the initial work we are reviewing references the article that has created the first draft for the AIF, but does not enter into many details. We would like to note the more detailed reasons that led to the decision of creating this format, as specified in [1].

Previously, there has been a variety of attempts at creating an argument mark-up language, but they were designed to be used with specific tools in mind. This led to languages that are too strongly linked with the internal workings of such tools and fail to provide the necessary abstraction and freedom to exchange arguments for different kinds of systems. In addition, such early works towards a mark-up language have been aimed at developing an intuitive user experience, accommodating the construction of natural language arguments. Thus, they neglected formal logic and were therefore unsuitable for various multi-agent systems.

With these initial limitations taken into consideration, the main objectives of the *Argumentation Interchange Format* were to, firstly, standardize the communication format of reasoning multi-agent systems, consequently facilitating their creation, and secondly to define an efficient and abstract way of exchanging data needed for either manipulating or visually representing arguments.

5.2 Foundations of the AIF

Returning to the original work we were following, the author goes on to elaborate on the core concepts of AIF. We will make an effort to formally display them in what follows. The foundation of AIF specifies the construction of arguments using two types of “nodes”. These can be *information nodes* (also called *I-nodes*), or *scheme nodes* (or simply *S-nodes*). Evidently, these nodes are part of disjoint sets:

- $\mathcal{N}_I \subset \mathcal{N}$, the subset containing information nodes
- $\mathcal{N}_S \subset \mathcal{N}$, the subset containing scheme nodes

In the above, $\mathcal{N}$ represents the set of all nodes. The differentiation between the types of nodes is important. Information nodes represent information, or data, such as premises or conclusions. S-nodes, however, are used to represent certain relations between I-nodes through *application of schemes*, modelling *patterns of reasoning.* These schemes, belonging to set $\mathcal{S}$, are also divided into three types:

- $\mathcal{S}^R \subset \mathcal{S}$, the subset containing *rule of inference* schemes
- $\mathcal{S}^C \subset \mathcal{S}$, the subset containing *conflict* schemes
- $\mathcal{S}^P \subset \mathcal{S}$, the subset containing *preference* schemes

Schemes are, in essence, very similar to rules of inference, only expanded also to non-deductive inference. There is also a predicate, denoted $uses : \mathcal{N}_S \times \mathcal{S}$ which shows a specific instantiation, or usage, of a scheme. It is important to note the distinction between a scheme and its application: the former is a generic reasoning pattern, while the latter is an individual, specific usage of that particular reasoning. This is formally specified as a constraint, using the function "uses": $\forall n \in \mathcal{N}_S, \exists s \in \mathcal{S}$, such that $\text{uses}(n, s)$.

Furthermore, scheme *nodes* can be divided into three categories, themselves:

- $\mathcal{N}_S^{RA} \subset \mathcal{N}_S$, the subset containing rule of inference *application* nodes (*RA-nodes*)
- $\mathcal{N}_S^{CA} \subset \mathcal{N}_S$, the subset containing conflict *application* nodes (*CA-nodes*)
- $\mathcal{N}_S^{PA} \subset \mathcal{N}_S$, the subset containing preference *application* nodes (*PA-nodes*)

The AIF distinguishes them from simple schemes, through the notion of "application", which means a specific instantiation of a certain scheme.

Linking information nodes through such application nodes intuitively leads to a graph representation of an argument. However, the AIF specifies that edges can only connect an I-node to an S-node (and vice-versa), or an S-node to another S-node. The specification denotes edges outgoing from an S-node as *scheme edges*, which support conclusions derived through that S-node. Conclusions can be either I-nodes or S-nodes themselves. Additionally, edges outgoing from an I-node are named *data edges*, which funnel information to scheme applications. For more specific details on each possible type of edge (such as RA-node to RA-node, or PA-node to RA-node, etc.), we refer the readers to [1, Table 1]. With these in mind, Rahwan gives the definitions for an *argument network* and a *simple argument.*

Definition 5.2.1 (argument network) *An argument network Φ is a graph, consisting of:*

- *a set $\mathcal{N} = \mathcal{N}_I \cup \mathcal{N}_S$ of vertices*
- *a binary relation $\xrightarrow{edge}: \mathcal{N} \times \mathcal{N}$, representing edges, with the restriction that $\forall i \in \mathcal{N}_I, \forall j \in \mathcal{N}_I$, $\nexists (i, j) \in \xrightarrow{edge}$*

In essence, an argument network is a graph with vertices that are either I-nodes or S-nodes, which are linked with *edges*, with the condition that two vertices can only be linked through an edge if they're not both I-nodes.

Definition 5.2.2 (simple argument) *A simple argument, in a network Φ and schemes $\mathcal{S}$ is a tuple $\langle P, \tau, c \rangle$, where:*

- *$P \subseteq \mathcal{N}_I$ is a set of I-nodes, constituting the premises*
- *$\tau \in \mathcal{N}_S^{RA}$ is an* RA-node
- *$c \in \mathcal{N}_I$ is an I-node representing the conclusion, with the condition that $\tau \xrightarrow{edge} c$, uses$(\tau, s)$, $s \in \mathcal{S}$ and $\forall p \in P$ there is $p \xrightarrow{edge} \tau$*

Basically, a simple argument is a graph with vertices consisting of premises and a conclusion (I-nodes) and rule of inference application nodes (RA-nodes). The edges between them represent either the supplying of data for the RA-nodes (I-to-RA), or the support of a conclusion through a scheme application (RA-to-I).

Interesting to notice is that attacks (CA-nodes) and preferences (PA-nodes) are not part of the simple argument, as per this definition, even if they are part of the argument network where the simple argument is found. As an example for this, in [10, Figure 5.6], the author offers two propositional logic argument networks. Perhaps of interest is to note the clear difference between a scheme and its application – modus ponens is an inference rule in propositional logic and is a *scheme* in the argument network. The author instantiates this scheme twice, namely as MP_1 and MP_2, acting as RA-nodes.

5.3 Natural language arguments to AIF

As a contribution, we will give an example of two arguments using natural language and then model them according to AIF, to further demonstrate the flexibility and the abstract nature of the Argument Interchange Format. As the basis of the arguments, we will discuss the sun's UV effects on health. The argument (also illustrated in Figure 2) is as follows:

(P_1) The sun's UV helps produce Vitamin D in your body

(P_2) Vitamin D is good for your health

(C_1) Therefore, the sun's UV is good for your health

We label the premises in order, as P_1, P_2, and the conclusion, as C_1. We will use hypothetical syllogism as a scheme, instantiating it as a rule of inference application node (RA-node) labelled HS_1. Essentially, the statements can be simplified to be much closer to propositional logic, to demonstrate the valid use of hypothetical syllogism:

(P_1) if *sun's UV*, then *Vitamin D*

(P_2) if *Vitamin D*, then *good health*

(C_1) if *sun's UV*, then *good health*

Therefore, we can construct a simple argument A_1, as per Definition 5.2.2.

Example 5.3.1 (simple argument in natural language) *The tuple $A_1 = \langle\{P_1, P_2\}, HS_1, C_1\rangle$ is a simple argument in natural language, where $P_1, P_2 \in \mathcal{N}_I$ are I-nodes representing premises and $C_1 \in \mathcal{N}_I$ is an I-node representing the conclusion. $HS_1 \in \mathcal{N}_S^{RA}$ is a rule of inference application node (RA-node), that uses the hypothetical syllogism scheme from propositional logic.*

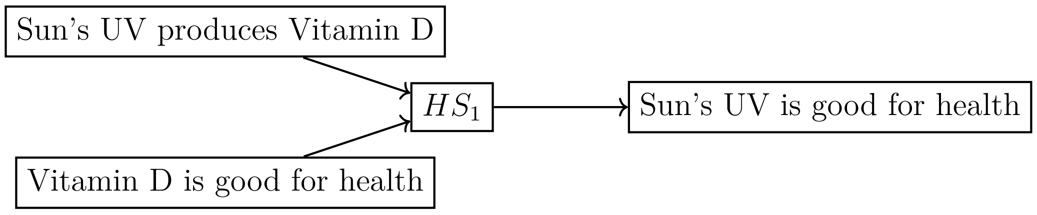
Additionally, we can come up with a rebuttal, such as:

(P_3) The sun's UV causes skin cancer

(P_4) Skin cancer is bad for your health

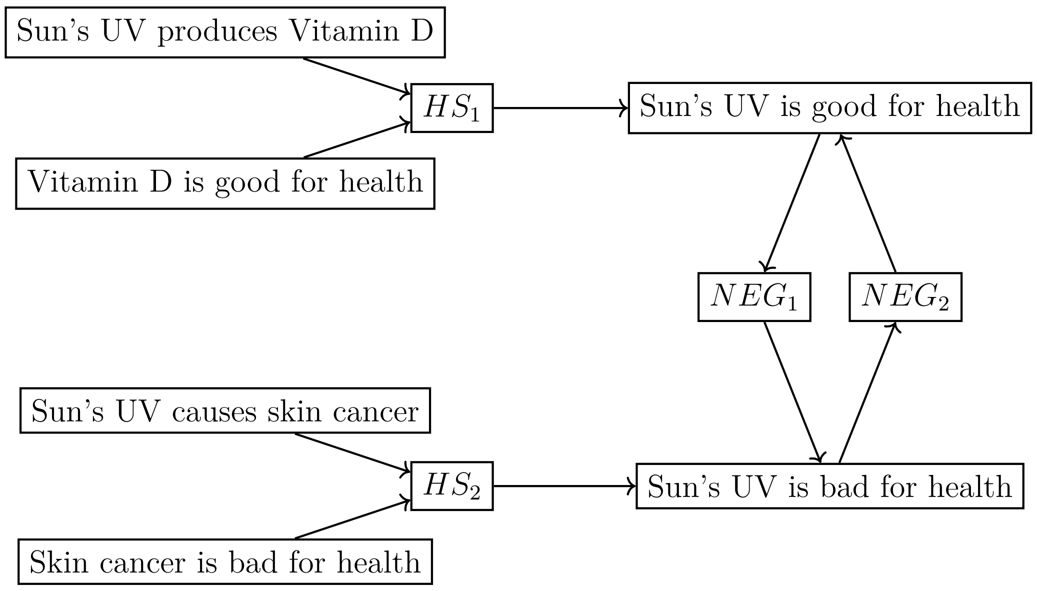
(C_2) Therefore, the sun's UV is bad for your health

After labelling this argument's premises with P_3, P_4 and its conclusion as C_2, we can represent the two arguments in an argument network (illustrated in Figure 3).



Sun's UV produces Vitamin D
HS_1
Sun's UV is good for health
Vitamin D is good for health

Figure 2: Argument network using natural language



Sun's UV produces Vitamin D
HS_1
Sun's UV is good for health
Vitamin D is good for health
NEG_1
NEG_2
Sun's UV causes skin cancer
HS_2
Sun's UV is bad for health
Skin cancer is bad for health

Figure 3: Argument network containing a rebuttal in natural language

Example 5.3.2 (rebuttal in natural language) *We use the previous simple argument $A_1 = \langle\{P_1, P_2\}, HS_1, C_1\rangle$ and similarly define another simple argument $A_2 = \langle\{P_3, P_4\}, HS_2, C_2\rangle$, where $P_3, P_4 \in \mathcal{N}_I$ are I-nodes representing premises and $C_2 \in \mathcal{N}_I$ is an I-node representing the conclusion. $HS_2 \in \mathcal{N}_S^{RA}$ is a rule of inference application node (RA-node), that uses the hypothetical syllogism scheme from propositional logic. It is clear that argument A_2 is a rebuttal to A_1, as the former's conclusion is the negation of the latter's conclusion. We display this conflict with CA-nodes NEG_1 and NEG_2, which are instantiations of a conflict scheme based on propositional contraries. We also note that, in natural language, "good health" is the negation of "bad health" and vice-versa.*

6 Conclusion

In the chapter we have explored, the author describes the core concepts of argumentation in multiagent systems and formalizes the basic theory on arguments. He continues with discussing different protocols for argumentation, as well as explaining strategic argumentation with the help of fundamental game theory. The chapter ends with elaboration on the Argument Interchange Format, a standardized way of representing and exchanging arguments.

In our work on reviewing and commentating this chapter, we have provided minor contributions, in the form of examples given by us. We have also tried to rephrase various formal definitions to clear possible misunderstandings, or provide additional explanations to some parts of the material.

References

[1] Carlos Chesñevar, Jarred McGinnis, Sanjay Modgil, Iyad Rahwan, Chris Reed, Guillermo Simari, Matthew South, Gerard Vreeswijk, and Steven Willmott. "Towards an Argument Interchange Format". In: *Knowl. Eng. Rev.* 21.4 (Dec. 2006), pp. 293–316. ISSN: 0269-8889. DOI: 10.1017/S0269888906001044. URL: https://doi.org/10.1017/S0269888906001044.

[2] Jacob Glazer and Ariel Rubinstein. "Debates and decisions: On a rationale of argumentation rules". In: *Games and Economic Behavior* 36.2 (2001), pp. 158–173.

[3] Nishan C Karunatillake, Nicholas R Jennings, Iyad Rahwan, and Peter McBurney. "Dialogue games that agents play within a society". In: *Artificial intelligence* 173.9-10 (2009), pp. 935–981.

[4] Sarit Kraus, Katia Sycara, and Amir Evenchik. "Reaching agreements through argumentation: a logical model and implementation". In: *Artificial Intelligence* 104.1-2 (1998), pp. 1–69.

[5] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory.* Vol. 1. Oxford university press New York, 1995.

[6] Sanjay Modgil and Martin Caminada. "Proof Theories and Algorithms for Abstract Argumentation Frameworks". In: *Argumentation in Artificial Intelligence.* Ed. by Guillermo Simari and Iyad Rahwan. Springer US, 2009, pp. 105–129. DOI: 10.1007/978-0-387-98197-0_6.

[7] Simon Parsons, Michael Wooldridge, and Leila Amgoud. "Properties and complexity of some formal inter-agent dialogues". In: *Journal of Logic and Computation* 13.3 (2003), pp. 347–376.

[8] Henry Prakken. “An abstract framework for argumentation with structured arguments”. In: *Argument & Computation* 1 (June 2010). DOI: 10.1080/19462160903564592.

[9] John Von Neumann and Oskar Morgenstern. “Theory of games and economic behavior”. In: *Theory of games and economic behavior*. Princeton university press, 2007.

[10] Gerhard Weiss. *Multiagent Systems. Second Edition*. MIT Press, 2013.