Argumentation among Agents: Review and Commentary

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

Introduction

- Reference: Iyad Rahwan's *Argumentation among Agents*, Chapter 5 in *Multiagent Systems*, by G. Weiss.

Introduction

- Reference: Iyad Rahwan's *Argumentation among Agents*, Chapter 5 in *Multiagent Systems*, by G. Weiss.
- Our contribution: several new examples, and proofs for some merely stated claims.

Introduction

- Reference: Iyad Rahwan's *Argumentation among Agents*, Chapter 5 in *Multiagent Systems*, by G. Weiss.
- Our contribution: several new examples, and proofs for some merely stated claims.
- What is the author attempting to formalize?

Introduction

- Reference: Iyad Rahwan's *Argumentation among Agents*, Chapter 5 in *Multiagent Systems*, by G. Weiss.
- Our contribution: several new examples, and proofs for some merely stated claims.
- What is the author attempting to formalize?
- The philosopher's view of argumentation: the giving of claims in favor or against a statement that is open for debate.

Prakken's framework, briefly

- Idea: generalize common logics by admitting two kinds of inference rules – *strict* and *defeasible*.

Prakken's framework, briefly

- Idea: generalize common logics by admitting two kinds of inference rules – *strict* and *defeasible*.
- An *argumentation system* is a tuple $(\mathcal{L}, cont, S, D)$.

Prakken's framework, briefly

- Idea: generalize common logics by admitting two kinds of inference rules – *strict* and *defeasible*.
- An *argumentation system* is a tuple $(\mathcal{L}, cont, S, D)$.
- $\mathcal{L}$ is some "logical language" (must contain $\neg$).

Prakken's framework, briefly

- Idea: generalize common logics by admitting two kinds of inference rules – *strict* and *defeasible*.
- An *argumentation system* is a tuple $(\mathcal{L}, cont, S, D)$.
- $\mathcal{L}$ is some "logical language" (must contain $\neg$).
- The function

$$cont : \mathcal{L} \to \mathcal{P}(\mathcal{L})$$

  generalizes negation.

Prakken's framework, briefly

- Idea: generalize common logics by admitting two kinds of inference rules – *strict* and *defeasible*.
- An *argumentation system* is a tuple $(\mathcal{L}, cont, S, D)$.
- $\mathcal{L}$ is some "logical language" (must contain $\neg$).
- The function

$$cont : \mathcal{L} \to \mathcal{P}(\mathcal{L})$$

  generalizes negation.
- S, D are respectively the sets of strict/defeasible inference rules.

Prakken's framework, briefly

- How does the *cont* function generalize negation?

Prakken's framework, briefly

- How does the *cont* function generalize negation?
- 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*.

Prakken's framework, briefly

- How does the *cont* function generalize negation?
- 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*.
- It is mandatory that

$$\neg\varphi \in cont(\varphi) \quad \text{and} \quad \varphi \in cont(\neg\varphi)$$

for any formula φ.

Prakken's framework, briefly

- An *argument* from a knowledge base $\mathcal{K}$ is defined similarly to a deduction in propositional logic. (The members of $\mathcal{K}$ play the role of the hypotheses.)

Prakken's framework, briefly

- An *argument* from a knowledge base $\mathcal{K}$ is defined similarly to a deduction in propositional logic. (The members of $\mathcal{K}$ play the role of the hypotheses.)
- Major difference: incorporation of the used inference rules.

Prakken's framework, briefly

- An *argument* from a knowledge base $\mathcal{K}$ is defined similarly to a deduction in propositional logic. (The members of $\mathcal{K}$ play the role of the hypotheses.)
- Major difference: incorporation of the used inference rules.
- The complete framework contains a partial order on defeasible rules. Using it, arguments may be compared.

Dung's model

- Henceforth, an *argumentation framework* will mean a finite directed graph $(\mathcal{A}, \rightharpoonup)$, whose nodes are called "arguments". The adjacency relation is pronounced "defeats".

Dung's model

- Henceforth, an *argumentation framework* will mean a finite directed graph $(\mathcal{A}, \rightharpoonup)$, whose nodes are called "arguments". The adjacency relation is pronounced "defeats".
- Hence, for arguments p, q, "$p \rightharpoonup q$" means "p defeats q".

Dung's model

- Henceforth, an *argumentation framework* will mean a finite directed graph $(\mathcal{A}, \rightharpoonup)$, whose nodes are called "arguments". The adjacency relation is pronounced "defeats".
- Hence, for arguments p, q, "$p \rightharpoonup q$" means "p defeats q".
- Note how the structure of arguments is not taken into account anymore.

Dung's model

- Henceforth, an *argumentation framework* will mean a finite directed graph $(\mathcal{A}, \rightharpoonup)$, whose nodes are called "arguments". The adjacency relation is pronounced "defeats".
- Hence, for arguments p, q, "$p \rightharpoonup q$" means "p defeats q".
- Note how the structure of arguments is not taken into account anymore.
- Objective: define an "acceptable" argument.

Dung's model

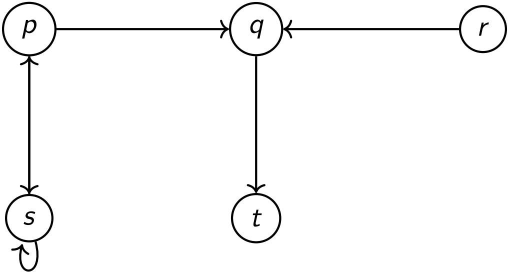

Figure: Our argumentation framework.

- $S^+ =$ the set of arguments defeated by some member of S.

Dung's model

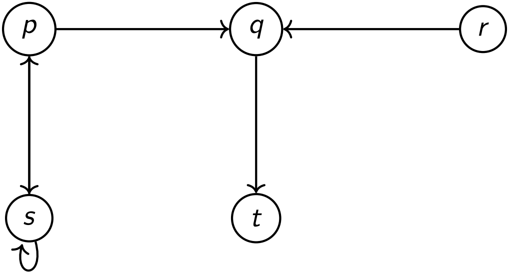

Figure: Our argumentation framework.

- $S^+ =$ the set of arguments defeated by some member of S.
- In the figure, $\{p, q\}^+ = \{q, s, t\}$.

Dung's model

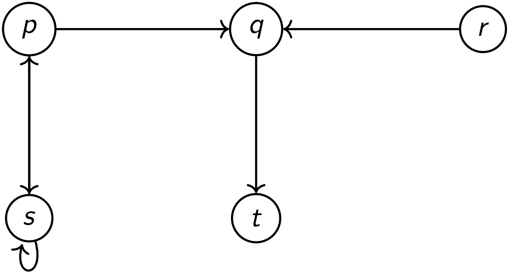

Figure: Our argumentation framework.

- $a^{-} =$ the set of arguments which defeat a.

Dung's model

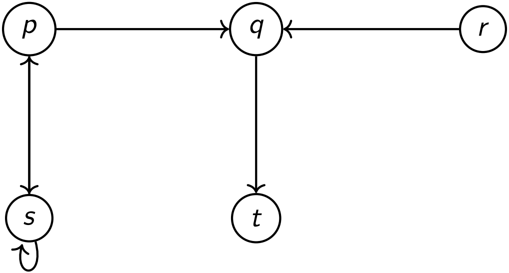

Figure: Our argumentation framework.

- $a^- =$ the set of arguments which defeat a.
- In the figure, $s^- = \{p, s\}$.

Dung's model

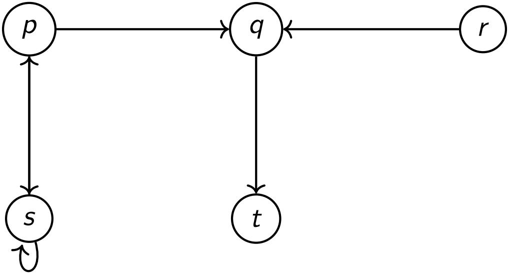

Figure: Our argumentation framework.

- A set S of arguments is *conflict-free* if no argument in S defeats another also in S.

Dung's model

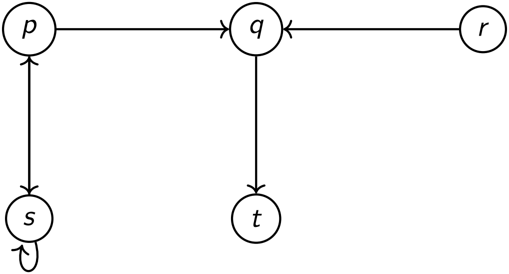

Figure: Our argumentation framework.

- A set S of arguments is *conflict-free* if no argument in S defeats another also in S.
- In the figure, $\{p, t\}$ and $\{r, t\}$ are conflict-free.

Dung's model

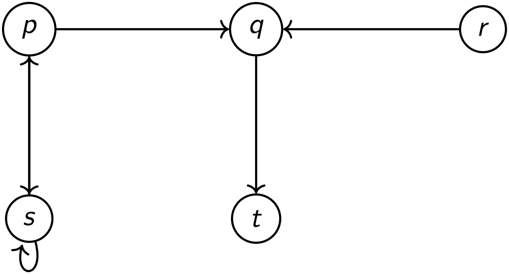

Figure: Our argumentation framework.

- A set S of arguments *defends* argument a if every argument which defeats a is defeated by S (i.e., is in S^+).

Dung's model

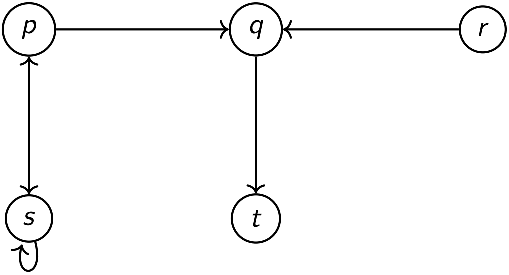

Figure: Our argumentation framework.

- A set S of arguments *defends* argument a if every argument which defeats a is defeated by S (i.e., is in S^+).
- In the figure, $\{p, t\}$ defends p.

Dung's model

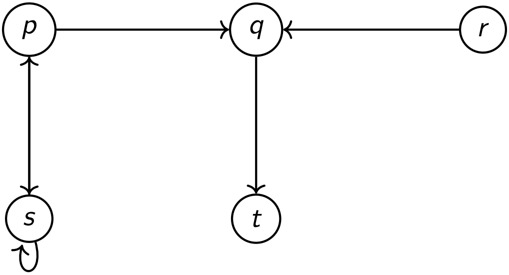

Figure: Our argumentation framework.

- The *characteristic function* $\mathcal{F}$ is defined thus:

$$\mathcal{F}(S) = \text{ the set of arguments defended by } S.$$

Dung's model

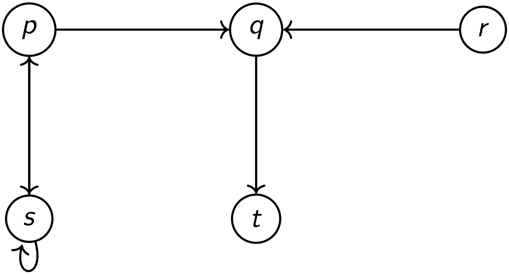

Figure: Our argumentation framework.

- The *characteristic function* $\mathcal{F}$ is defined thus:
  $\mathcal{F}(S) =$ the set of arguments defended by S.
- In the figure, $\mathcal{F}(\{p, q, r\}) = \{p, r, t\}$ and $\mathcal{F}(\{r, t\}) = \{r, t\}$.

Dung's model

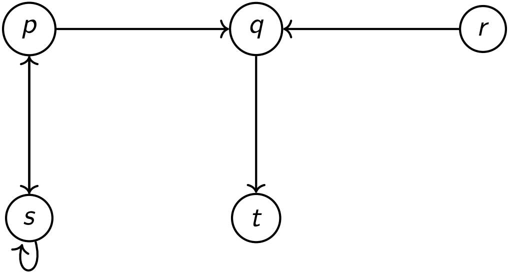

Figure: Our argumentation framework.

- A *complete extension* is a set S of arguments which is conflict-free and such that $\mathcal{F}(S) = S$ (i.e., it defends its own members and nothing else).

Dung's model

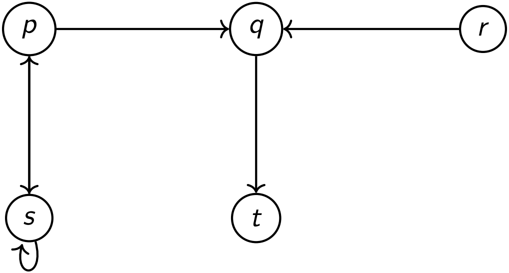

Figure: Our argumentation framework.

- A *complete extension* is a set S of arguments which is conflict-free and such that $\mathcal{F}(S) = S$ (i.e., it defends its own members and nothing else).
- By the remarks on previous slides, $\{r, t\}$ is a complete extension.

Dung's model

- The author exhibits an equivalent characterization of complete extensions via *labellings*.

Dung's model

- The author exhibits an equivalent characterization of complete extensions via *labellings*.
- 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.

Argumentation Games

- The author focuses on Dung's 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.

Argumentation Games

- The author focuses on Dung's 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.
- Objective: Formalize such an argumentation process and additionally enforce a set of constraints in order to capture various semantics (for example, an agent cannot contradict himself).

Argumentation Games

- Two players: **PRO** and **OPP**

Argumentation Games

- Two players: **PRO** and **OPP**
- PRO is the proponent who states the initial argument.

Argumentation Games

- Two players: **PRO** and **OPP**
- PRO is the proponent who states the initial argument.
- OPP is the opponent who begins by counter-attacking the argument proposed by PRO.

Argumentation Games

- Two players: **PRO** and **OPP**
- PRO is the proponent who states the initial argument.
- OPP is the opponent who begins by counter-attacking the argument proposed by PRO.
- Both players take turns in defeating the last argument that has been put forward by their counterpart player.

Argumentation Games

- Two players: **PRO** and **OPP**
- PRO is the proponent who states the initial argument.
- OPP is the opponent who begins by counter-attacking the argument proposed by PRO.
- Both players take turns in defeating the last argument that has been put forward by their counterpart player.
- The game is considered to be won by the player who states an argument a that cannot be defeated (i.e. $a^- = \emptyset$)

What is a dispute?

- The author calls the sequence of moves done by the players a *dispute*, a notion which is intensively used in further definitions and proofs.

What is a dispute?

- The author calls the sequence of moves done by the players a *dispute*, a notion which is intensively used in further definitions and proofs.
- However, a concrete definition is not provided. We attempt to state the following formal definition:

What is a dispute?

- The author calls the sequence of moves done by the players a *dispute*, a notion which is intensively used in further definitions and proofs.
- However, a concrete definition is not provided. We attempt to state the following formal definition:

Definition (dispute)

Given an argumentation framework $(\mathcal{A}, \rightharpoonup)$, a dispute is a nonempty, possibly infinite sequence d of arguments in $\mathcal{A}$ with the following property: $d_{i+1} \rightharpoonup d_i$, whenever i and $i+1$ are in d's domain (i.e. every argument in the sequence defeats its preceding argument).

Dispute trees

- Observation: A player could potentially counter-attack its counterpart player with any argument whatsoever that defeats the last move.

Dispute trees

- Observation: 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.

Dispute trees

- Observation: 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.
- Again, a definition is not provided. We attempt to adapt one from *Modgil et al.*

Dispute trees

- Observation: 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.
- Again, a definition is not provided. We attempt to adapt one from *Modgil et al.*

Definition (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 is labelled with an argument in $\mathcal{A}$ and for every node v, v has a child labelled x iff v's label is defeated by x.

An example from the book

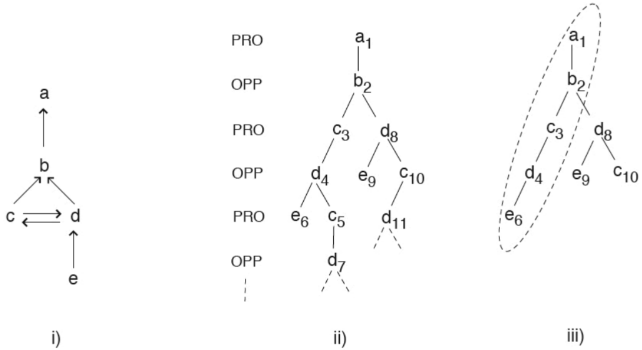

Figure 5.3: Argumentation framework and dispute tree. (i) shows an argumentation framework, (ii) shows the dispute tree induced in a, and (iii) shows the dispute tree induced by a under protocol G, with the winning strategy encircled.

Protocol G

- The author establishes a rule (called protocol G) by which the PRO player cannot repeat an argument in a dispute.

Protocol G

- The author establishes a rule (called protocol G) by which the PRO player cannot repeat an argument in a dispute.
- We provide a definition with regard to a dispute tree:

Protocol G

- The author establishes a rule (called protocol G) by which the PRO player cannot repeat an argument in a dispute.
- We provide a definition with regard to a dispute tree:

Definition (dispute tree under protocol G)

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

Protocol G

- The author establishes a rule (called protocol G) by which the PRO player cannot repeat an argument in a dispute.
- We provide a definition with regard to a dispute tree:

Definition (dispute tree under protocol G)

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

- It is claimed by the author that the following property is true, to which we provide a proof:

Protocol G

- The author establishes a rule (called protocol G) by which the PRO player cannot repeat an argument in a dispute.
- We provide a definition with regard to a dispute tree:

Definition (dispute tree under protocol G)

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

- It is claimed by the author that the following property is true, to which we provide a proof:

Claim

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

Proof sketch

- Let $n = card(\mathcal{A})$ and d 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).

Proof sketch

- Let $n = card(\mathcal{A})$ and d 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 d_{2n-1} from our sequence d.

Proof sketch

- Let $n = card(\mathcal{A})$ and d 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 d_{2n-1} from our sequence d.
- By protocol G, we have exactly n different arguments uttered by PRO, which cover all the arguments in the set $\mathcal{A}$.

Proof sketch

- Let $n = card(\mathcal{A})$ and d 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 d_{2n-1} from our sequence d.
- By protocol G, we have exactly n different arguments uttered by PRO, which cover all the arguments in the set $\mathcal{A}$.
- Assume that d_{2n} exists. This being the $n+1$'st argument stated by PRO, it must coincide with an argument that has been uttered before, since we have a total of only n different arguments to choose from (Dirichlet's box principle).

Proof sketch

- Let $n = card(\mathcal{A})$ and d 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 d_{2n-1} from our sequence d.
- By protocol G, we have exactly n different arguments uttered by PRO, which cover all the arguments in the set $\mathcal{A}$.
- Assume that d_{2n} exists. This being the $n+1$'st argument stated by PRO, it must coincide with an argument that has been uttered before, since we have a total of only n different arguments to choose from (Dirichlet's box principle).
- However, this contradicts protocol G. Hence, d is finite.

Strategic Argumentation & Game Theory

- Background on the analysis of strategic argumentation
- Why Game Theory
- Important Game Theory Concepts

Overview of strategic argumentation

Overview of strategic argumentation

- Various argumentation systems introduced. Each defines restrictions regarding what agents can and cannot do (e.g. Prakken's framework)

Overview of strategic argumentation

- Various argumentation systems introduced. Each defines restrictions regarding what agents can and cannot do (e.g. Prakken's framework)
- Behaviour of agents must be analyzed. This is called a "strategy"

Overview of strategic argumentation

- Various argumentation systems introduced. Each defines restrictions regarding what agents can and cannot do (e.g. Prakken's framework)
- Behaviour of agents must be analyzed. This is called a "strategy"
- Parsons et al. introduce the *dialogue system* based on *attitudes* (e.g. confident, careful, thoughtful)

Overview of strategic argumentation

- Various argumentation systems introduced. Each defines restrictions regarding what agents can and cannot do (e.g. Prakken's framework)
- Behaviour of agents must be analyzed. This is called a "strategy"
- Parsons et al. introduce the *dialogue system* based on *attitudes* (e.g. confident, careful, thoughtful)
- Other models based on *social constructs* or *mental states* are proposed by Nishan et al. and Kraus et al.

Why Game Theory

Why Game Theory

- Previous euristic approaches only consider a subset of all strategies

Why Game Theory

- Previous euristic approaches only consider a subset of all strategies
- Game Theory provides a framework appropritate for a comprehensive analysis of *strategic argumentation*. It can be used for:

Why Game Theory

- Previous euristic approaches only consider a subset of all strategies
- Game Theory provides a framework appropritate for a comprehensive analysis of *strategic argumentation*. It can be used for:
  - Predicting the outcome of a specific scenario

Why Game Theory

- Previous euristic approaches only consider a subset of all strategies
- Game Theory provides a framework appropritate for a comprehensive analysis of *strategic argumentation*. It can be used for:
  - Predicting the outcome of a specific scenario
  - Designing a protocol such that a set of known agents behave in a desireable way (called **mechanism design**)

Glazer & Rubenstein's Model

- One of the first attempts of analyzing argumentation based on game theory
- *Procedural rules* (order and type of arguments) and *persuation rules* (how the outcome is chosen / who wins the debate)
- No correlation between the logical structure of the information presented and the choice of the outcome

Game Theory Concepts

Game Theory Concepts

- I is the set of self-interested agents

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i
- $o \in \mathcal{O}$ is an outcome

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i
- $o \in \mathcal{O}$ is an outcome
- utility function $u(o, \theta_i)$ defines *how much* agent i prefers outcome o

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i
- $o \in \mathcal{O}$ is an outcome
- utility function $u(o, \theta_i)$ defines *how much* agent i prefers outcome o
- $s(\theta_i) \in \Sigma_i$ is a *strategy* of agent i

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i
- $o \in \mathcal{O}$ is an outcome
- utility function $u(o, \theta_i)$ defines *how much* agent i prefers outcome o
- $s(\theta_i) \in \Sigma_i$ is a *strategy* of agent i
- $s = (s_1(\theta_1), \ldots, s_I(\theta_I)) \in \mathcal{O}$ is a *strategy profile*

Game Theory Concepts

- I is the set of self-interested agents
- $\theta_i \in \Theta$ is the type of agent i
- $o \in \mathcal{O}$ is an outcome
- utility function $u(o, \theta_i)$ defines *how much* agent i prefers outcome o
- $s(\theta_i) \in \Sigma_i$ is a *strategy* of agent i
- $s = (s_1(\theta_1), \ldots, s_I(\theta_I)) \in \mathcal{O}$ is a *strategy profile*
- for convenience:
  $s_{-i}(\theta_{-i}) = (s_1(\theta_1), \ldots, s_{i-1}(\theta_{i-1}), s_{i+1}(\theta_{i+1}), \ldots, s_I(\theta_I))$
  $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_I)$

Solution Concepts

Solution Concepts

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

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

Solution Concepts

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.

Solution Concepts

Let $s^* = (s_1^*, \dots, s_I^*)$ 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.

Problems:

Solution Concepts

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.

Problems:

- Can be multple Nash equilibria

Solution Concepts

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.

Problems:

- Can be multple Nash equilibria
- Perfect knowledge of agent types is assumed

Solution Concepts

Solution Concepts

Formally, a strategy $s_i^* \in \Sigma_i$ is said to be *dominant* if

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

Solution Concepts

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.

Solution Concepts

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.

Compared to the *Nash equilibrium*, it is more solid as no information about other agents needs to be assumed.

Solution Concepts

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.

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.

Mechanism Design

Mechanism Design

The problem studied by mechanism design is that of achieving a desired outcome when dealing with a group of self-interested agents.

Mechanism Design

The problem studied by mechanism design is that of achieving a desired outcome when dealing with a group of self-interested agents.

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

Mechanism Design

The problem studied by mechanism design is that of achieving a desired outcome when dealing with a group of self-interested agents.

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 probleme with it that it is based on private information of the agents (type). Agents cannot be trusted to be truthful.

Mechanism Design

Mechanism Design

- $\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is a restricted set of strategies that agents can choose from

Mechanism Design

- $\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is a restricted set of strategies that agents can choose from
- Σ_i is the set of strategies that agent i choosing from

Mechanism Design

- $\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is a restricted set of strategies that agents can choose from
- Σ_i is the set of strategies that agent i choosing from
- $g : \Sigma \rightarrow \mathcal{O}$ is a function that matches strategy profiles to outcomes

Mechanism Design

- $\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is a restricted set of strategies that agents can choose from
- Σ_i is the set of strategies that agent i choosing from
- $g : \Sigma \to \mathcal{O}$ is a function that matches strategy profiles to outcomes
- $\mathcal{M} = (\Sigma, g(\cdot))$ is called a mechanism.

Mechanism Design

- $\Sigma = \Sigma_1 \times \cdots \times \Sigma_I$ is a restricted set of strategies that agents can choose from
- Σ_i is the set of strategies that agent i choosing from
- $g : \Sigma \to \mathcal{O}$ is a function that matches strategy profiles to outcomes
- $\mathcal{M} = (\Sigma, g(\cdot))$ is called a mechanism.

A mechanism is said to define a game where the strategy choices of the agents are limited to Σ. To maximize its utility, agent i can only choose strategies from Σ_i.

Mechanism Design

Mechanism Design

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

Mechanism Design

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

Informally, a 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.

Mechanism Design

Mechanism Design

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 θ_i' to the *mechanism*.

Mechanism Design

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 θ_i' to the *mechanism*.

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.

Revelation Principle

Revelation Principle

The definition of the mechanism states that the search space for strategies is infinite.

Revelation Principle

The definition of the mechanism states that the search space for strategies is infinite.

The *Revelation Principle* helps limit the search-space and states that:

Revelation Principle

The definition of the mechanism states that the search space for strategies is infinite.

The *Revelation Principle* helps limit the search-space and states that:

If there exists some mechanism that implements social choice function f in dominant strategies, then there exists a direct-revealing mechanism that implements f in dominant strategies and is truthful.

The Argument Interchange Format

- Stems from a need for standardized representations of arguments.

The Argument Interchange Format

- Stems from a need for standardized representations of arguments.
- Previous attempts unsuitable:

The Argument Interchange Format

- Stems from a need for standardized representations of arguments.
- Previous attempts unsuitable:
  - designed to be used with specific tools

The Argument Interchange Format

- Stems from a need for standardized representations of arguments.
- Previous attempts unsuitable:
  - designed to be used with specific tools
  - strong link between language and tool

The Argument Interchange Format

- Stems from a need for standardized representations of arguments.
- Previous attempts unsuitable:
  - designed to be used with specific tools
  - strong link between language and tool
  - neglected formal logic due to user experience focus

The Argument Interchange Format

- Objectives:

The Argument Interchange Format

- Objectives:
  - standardize communication between reasoning-based multi-agent systems

The Argument Interchange Format

- Objectives:
  - standardize communication between reasoning-based multi-agent systems
  - facilitate the creation of such systems

The Argument Interchange Format

- Objectives:
  - standardize communication between reasoning-based multi-agent systems
  - facilitate the creation of such systems
  - design an efficient and abstract format for exchanging data

The Argument Interchange Format

- Objectives:
  - standardize communication between reasoning-based multi-agent systems
  - facilitate the creation of such systems
  - design an efficient and abstract format for exchanging data
  - facilitate argument manipulation and visual representation

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)
  - $\mathcal{N}_S \subset \mathcal{N}$, scheme nodes (S-nodes)
- Schemes are classes of reasoning patterns.

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)
  - $\mathcal{N}_S \subset \mathcal{N}$, scheme nodes (S-nodes)
- Schemes are classes of reasoning patterns.
- Schemes are divided into:

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)
  - $\mathcal{N}_S \subset \mathcal{N}$, scheme nodes (S-nodes)
- Schemes are classes of reasoning patterns.
- Schemes are divided into:
  - $\mathcal{S}^R \subset \mathcal{S}$, *rule of inference* schemes

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)
  - $\mathcal{N}_S \subset \mathcal{N}$, scheme nodes (S-nodes)
- Schemes are classes of reasoning patterns.
- Schemes are divided into:
  - $\mathcal{S}^R \subset \mathcal{S}$, *rule of inference* schemes
  - $\mathcal{S}^C \subset \mathcal{S}$, *conflict* schemes

Foundational concepts of the AIF

- Arguments are composed of networks of interlinked nodes.
- Two types of nodes:
  - $\mathcal{N}_I \subset \mathcal{N}$, information nodes (I-nodes)
  - $\mathcal{N}_S \subset \mathcal{N}$, scheme nodes (S-nodes)
- Schemes are classes of reasoning patterns.
- Schemes are divided into:
  - $\mathcal{S}^R \subset \mathcal{S}$, *rule of inference* schemes
  - $\mathcal{S}^C \subset \mathcal{S}$, *conflict* schemes
  - $\mathcal{S}^P \subset \mathcal{S}$, *preference* schemes

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.
- S-nodes are of three types, for each scheme class:

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.
- S-nodes are of three types, for each scheme class:
  - $\mathcal{N}_S^{RA} \subset \mathcal{N}_S$, rule of inference *application* nodes (*RA-nodes*)

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.
- S-nodes are of three types, for each scheme class:
  - $\mathcal{N}_S^{RA} \subset \mathcal{N}_S$, rule of inference *application* nodes (*RA-nodes*)
  - $\mathcal{N}_S^{CA} \subset \mathcal{N}_S$, conflict *application* nodes (*CA-nodes*)

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.
- S-nodes are of three types, for each scheme class:
  - $\mathcal{N}_S^{RA} \subset \mathcal{N}_S$, rule of inference *application* nodes (*RA-nodes*)
  - $\mathcal{N}_S^{CA} \subset \mathcal{N}_S$, conflict *application* nodes (*CA-nodes*)
  - $\mathcal{N}_S^{PA} \subset \mathcal{N}_S$, preference *application* nodes (*PA-nodes*)

Foundational concepts of the AIF

- S-nodes are actual *applications* of a scheme.
- S-nodes are of three types, for each scheme class:
  - $\mathcal{N}_S^{RA} \subset \mathcal{N}_S$, rule of inference *application* nodes (*RA-nodes*)
  - $\mathcal{N}_S^{CA} \subset \mathcal{N}_S$, conflict *application* nodes (*CA-nodes*)
  - $\mathcal{N}_S^{PA} \subset \mathcal{N}_S$, preference *application* nodes (*PA-nodes*)
- Example: $MP_1 \in \mathcal{N}_S^{RA}$, an RA-node implementing the modus ponens *rule of inference scheme* from propositional logic.

Argument network

- An argument network Φ is a graph, consisting of:

Argument network

- An argument network Φ is a graph, consisting of:
  - a set $\mathcal{N} = \mathcal{N}_I \cup \mathcal{N}_S$ of vertices

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 $\forall i \in \mathcal{N}_I, \forall j \in \mathcal{N}_I, \nexists (i,j) \in \xrightarrow{edge}$

A simple argument

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

A 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

A 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*

A 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 $p \xrightarrow{edge} \tau$

Natural language arguments to AIF

- The argument:

Natural language arguments to AIF

- The argument:

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

Natural language arguments to AIF

- The argument:

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

(P_2) Vitamin D is good for your health

Natural language arguments to AIF

- The argument:

(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

Natural language arguments to AIF

- The argument:
  - (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 construct the tuple $A_1 = \langle \{P_1, P_2\}, HS_1, C_1 \rangle$, a simple argument in natural language, where $P_1, P_2 \in \mathcal{N}_I$ are premises and $C_1 \in \mathcal{N}_I$ is the conclusion. $HS_1 \in \mathcal{N}_S^{RA}$ is an RA-node, that uses the hypothetical syllogism scheme from propositional logic.

Natural language arguments to AIF

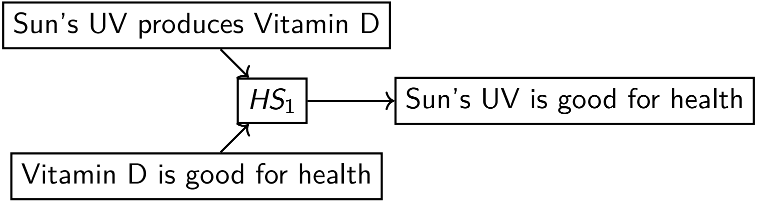

Figure: Argument network using natural language

Natural language arguments to AIF

- Coming up with a rebuttal:

Natural language arguments to AIF

- Coming up with a rebuttal:

(P_3) The sun's UV causes skin cancer

Natural language arguments to AIF

- Coming up with a rebuttal:

(P_3) The sun's UV causes skin cancer
(P_4) Skin cancer is bad for your health

Natural language arguments to AIF

- Coming up with a rebuttal:

(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

Natural language arguments to AIF

- Coming up with a rebuttal:
  - (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
- 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 premises and $C_2 \in \mathcal{N}_I$ is the conclusion. $HS_2 \in \mathcal{N}_S^{RA}$ is an RA-node, that uses the hypothetical syllogism scheme from propositional logic. Conflict is displayed with CA-nodes NEG_1 and NEG_2, instantiations of a conflict scheme based on propositional contraries.

Natural language arguments to AIF

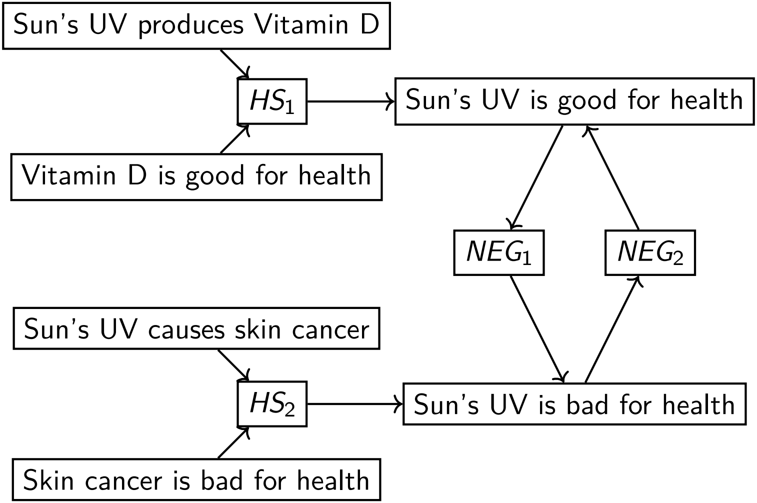

Figure: Argument network containing a rebuttal in natural language