Lecture 7

Abstract Argumentation Frameworks

INST0074

Lecture Outline

- Abstract Argumentation Frameworks
  - Main Ideas & Definitions
- Acceptability Semantics
  - Extension-based semantics
  - Labelling-based semantics

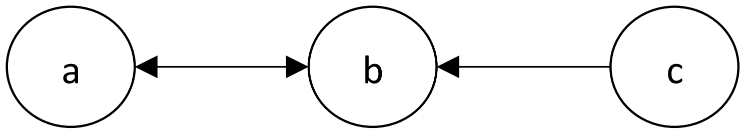
AAFs: Main Ideas

- Dung, P.M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. Artificial Intelligence, 77:321-357, 1995.
- Arguments are defeasible entities that may attack each other
- The acceptance of an argument depends only on the status of the arguments that attack it.
- The structure, the origin and any other information about the arguments are abstracted away.
- Acceptability semantics formally define which arguments are accepted and which are rejected.

AAFs: Definitions

- An **argumentation framework** is a directed graph, the nodes of which are **arguments**, whereas the edges represent **attacks** among the arguments.
- $\mathbf{AF = \{A, R\}, R \subseteq A \times A}$
  - $\mathbf{A}$ is a set of arguments
  - $\mathbf{R}$ is a binary relation on A
  - If $\mathbf{(a, b)} \in \mathbf{R}$ then we say that **a attacks b**
  - A set of arguments $\mathbf{S} \subseteq \mathbf{A}$ attacks an argument $\mathbf{b} \in \mathbf{A}$ iff there is an argument $\mathbf{a} \in \mathbf{S}$ that attacks $\mathbf{b}$
  - A set of arguments $\mathbf{S} \subseteq \mathbf{A}$ is **conflict-free** iff there are no arguments $\mathbf{a, b} \in \mathbf{S}$ such that $\mathbf{a}$ attacks $\mathbf{b}$

AAFs: An example



Argumentation Framework

AF = {A, R}
A = {a, b, c}
R = {(a,b),(b,a),(c,b)}

Attacks by sets of arguments

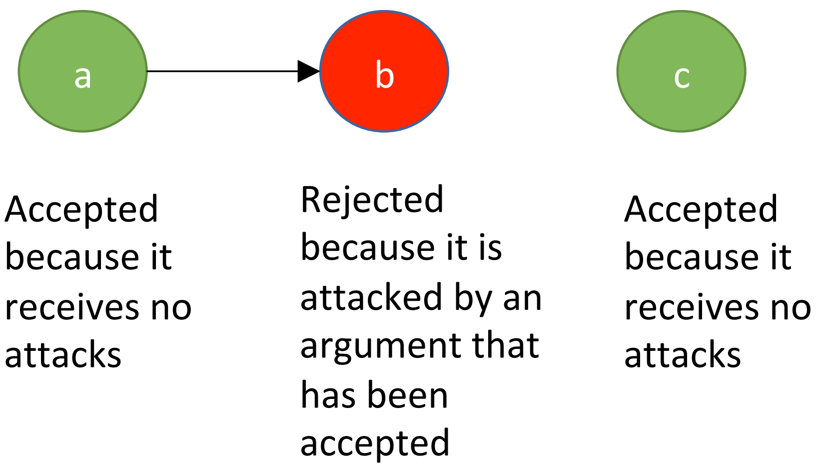
{a} attacks b
{b} attacks a
{c} attacks b
{a, b} attacks a
{a, b} attacks b
{a, c} attacks b
{b, c} attacks a
{b, c} attacks b
{a, b, c} attacks a
{a, b, c} attacks b

Conflict-free sets of arguments

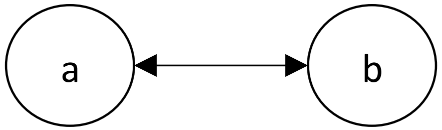
{}, {a}, {b}, {c}, {a,c}

Evaluation of arguments

- An argument is accepted if it does not receive any attacks.
- An argument is rejected if there is a counter-argument that has been accepted.
- An argument that does not attack and is not attacked by any other argument does not affect the acceptability of the other arguments.



A more complex case



- Arguments that are in conflict cannot be both accepted
- Should we accept neither or either of them?
- Scenario 1:
  - a: The weather in Cuba is great, let's go there for our holidays.
  - b: The tickets to Cuba are expensive, let's go somewhere else.

Accept either

- Scenario 2:
  - a: Alice: Bob committed the murder. I was him in the crime scene.
  - b: Bob: I didn't do it. Alice did it. She hated the victim!

Accept neither

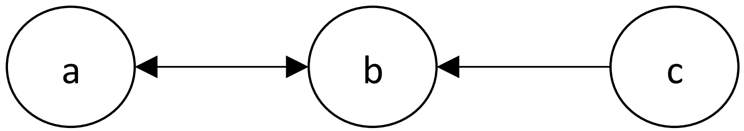
Extension-based acceptability semantics

- The acceptability of arguments can be defined using the notion of extensions.
- An **extension** of an argumentation framework $\mathbf{AF} = \{\mathbf{A}, \mathbf{R}\}$ is a set of arguments $\mathbf{E} \subseteq \mathbf{A}$ that we can reasonably accept.
- An **extension-based semantics** provides a formal way for identifying extensions (i.e. selecting sets of arguments that are reasonable to accept), according to some criterion.

Admissibility

- The notion of admissible sets of arguments can be regarded as the minimum requirement for a set of arguments to be accepted.
- A set of arguments $\mathbf{S} \subseteq \mathbf{A}$ **defends an argument** $\mathbf{a} \in \mathbf{A}$ iff it attacks any argument $\mathbf{b} \in \mathbf{A}$ that attacks a
- A set of arguments $\mathbf{E} \subseteq \mathbf{A}$ is **admissible** iff it is conflict-free and defends all its elements.

Admissibility (example)

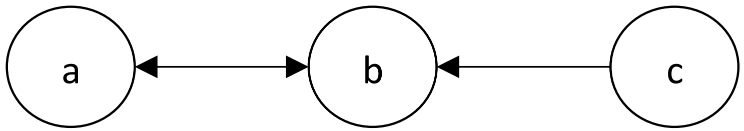


Conflict-free	Admissible	
{}	✓	It receives no attacks
{a}	✓	It defends itself from b
{b}	✘	It doesn’t defend itself from c
{c}	✓	It receives no attacks
{a,c}	✓	It defends itself from b

Complete semantics

- Complete semantics is based on the notion of admissibility
  - A complete extension must be an admissible set of arguments
- It additionally requires accepting any argument that can be defended by an admissible set of arguments
- A set of arguments $\mathbf{E} \subseteq \mathbf{A}$ is **a complete extension** of $\mathbf{AF} = \{\mathbf{A}, \mathbf{R}\}$ iff it is admissible and contains all the arguments it defends

Complete semantics (example)



Conflict-free	Admissible	Complete	
{}	✓	✘	It defends c but doesn’t contain it
{a}	✓	✘	It defends c but doesn’t contain it
{b}	✘	✘	It is not admissible
{c}	✓	✘	It defends a but doesn’t contain it
{a,c}	✓	✓	It contains all the arguments it defends

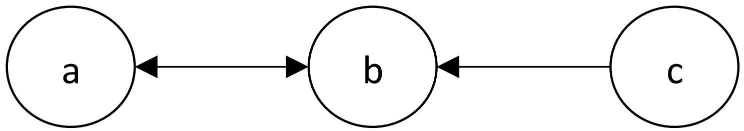
Grounded semantics

- The most conservative (sceptical) semantics regarding the number of arguments it accepts.
- It accepts only the arguments we cannot avoid to accept
- A set of arguments $\mathbf{E} \subseteq \mathbf{A}$ is **a grounded extension** of $\mathbf{AF} = \{\mathbf{A}, \mathbf{R}\}$ iff it is the minimal (w.r.t. set inclusion) complete extension of $\mathbf{AF}$

Minimal and maximal sets

- If $\mathbf{S}$ is set of sets
  - A set $\mathbf{X} \in \mathbf{S}$ is minimal iff there is no set $\mathbf{Y} \in \mathbf{S}$ such that $\mathbf{Y} \subset \mathbf{X}$
  - A set $\mathbf{X} \in \mathbf{S}$ is maximal iff there is no set $\mathbf{Y} \in \mathbf{S}$ such that $\mathbf{X} \subset \mathbf{Y}$
- For example, if $\mathrm{S} = \{\{e\}, \{a,b\}, \{a,c\}, \{a,b,c\}, \{a,b,d\}, \{a,b,e\}, \{a,b,c,e\}\}$
  - The minimal sets are: $\{e\}$, $\{a,b\}$, $\{a,c\}$
  - The maximal sets are: $\{a,b,d\}$, $\{a,b,c,e\}$

Grounded semantics (example)



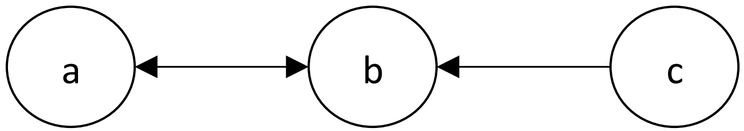
Conflict-free	Admissible	Complete	Grounded
{}	✓	✘	✘
{a}	✓	✘	✘
{b}	✘	✘	✘
{c}	✓	✘	✘
{a,c}	✓	✓	✓

The only complete extension is also a grounded extension

Preferred semantics

- The most credulous semantics.
- It accepts as many arguments as possible
- A set of arguments $\mathbf{E} \subseteq \mathbf{A}$ is **a preferred extension** of $\mathbf{AF} = \{\mathbf{A}, \mathbf{R}\}$ iff it is a maximal (w.r.t. set inclusion) complete extension of $\mathbf{AF}$

Preferred semantics (example)



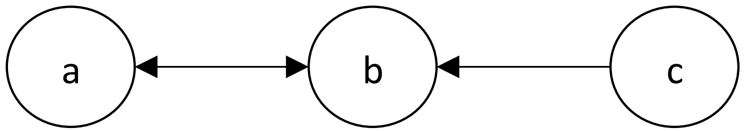
Conflict-free	Admissible	Complete	Grounded	Preferred
{}	✓	✘	✘	✘
{a}	✓	✘	✘	✘
{b}	✘	✘	✘	✘
{c}	✓	✘	✘	✘
{a,c}	✓	✓	✓	✓

The only complete extension is also a preferred extension

Stable semantics

- It requires that every argument is either accepted or attacked by an accepted argument (and is therefore rejected).
- A set of arguments $\mathbf{E} \subseteq \mathbf{A}$ is **a stable extension** of $\mathbf{AF} = \{\mathbf{A}, \mathbf{R}\}$ iff it is conflict-free and attacks all arguments in $\mathbf{A} / \mathbf{E}$

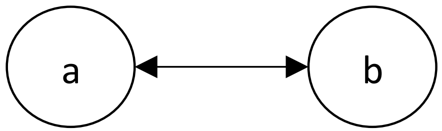
Stable semantics (example)



Conflict-free	Admissible	Complete	Grounded	Preferred	Stable
{}	✓	✘	✘	✘	✘
{a}	✓	✘	✘	✘	✘
{b}	✘	✘	✘	✘	✘
{c}	✓	✘	✘	✘	✘
{a,c}	✓	✓	✓	✓	✓

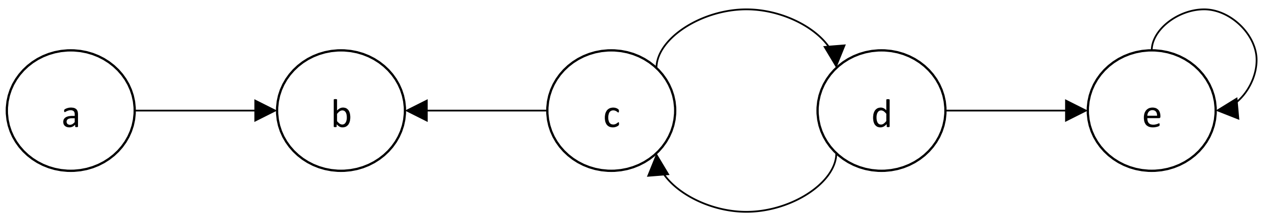
It attacks the arguments it doesn't contain.

More examples



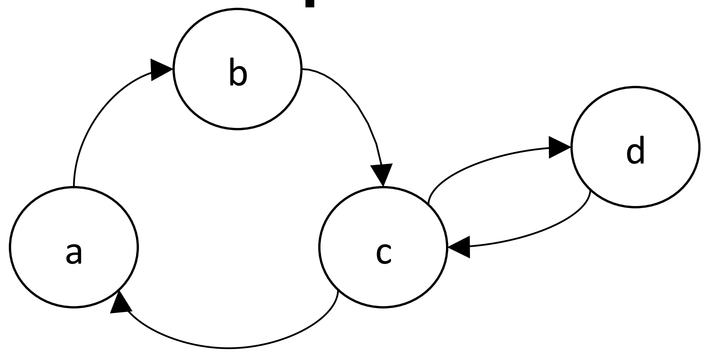
Admissible	Complete	Grounded	Preferred	Stable
{}	✓	✓	✘	✘
{a}	✓	✘	✓	✓
{b}	✓	✘	✓	✓

More examples



Admissible	Complete	Grounded	Preferred	Stable
{}	✘	✘	✘	✘
{a}	✓	✓	✘	✘
{a,c}	✓	✘	✓	✘
{a,d}	✓	✘	✓	✓
{c}	✘	✘	✘	✘
{d}	✘	✘	✘	✘

More examples



Admissible	Complete	Grounded	Preferred	Stable
{}	✓	✓	✘	✘
{d}	✘	✘	✘	✘
{a,d}	✓	✘	✓	✓

Properties of extensions

- The empty set is always admissible.
- There is always a preferred extension.
- The grounded extension is the intersection of all complete extensions and is unique.
- No stable extension is empty but there are argument frameworks for which there is no stable extension.
  - Consider for example this: **A** = **{a, b, c}**, **R** = **{(a,b),(b,c),(c,a)}**
- Every stable extension is also a preferred extension.
- If an argument graph has no cycle then there is a single extension that is stable, preferred, complete and grounded.

Labelling-based acceptability semantics

- Each argument in the framework is assigned a **label**:
  - **Lab(a)** = **in**: the argument is accepted
  - **Lab(a)** = **out**: the argument is rejected
  - **Lab(a)** = **undec**: the argument is neither accepted nor rejected
- A **labelling-based semantics** provides a way to select "reasonable" labellings among all the possible ones, according to some criterion.
- Legal labellings
  - An **in**-labelled argument is said to be **legally in** iff all its attackers are labelled **out**.
  - An **out**-labelled argument is said to be **legally out** iff at least one of its attackers is labelled **in**.
  - An **undec**-labelled argument is said to be **legally undec** iff not all its attackers are labelled **out** and it doesn't have an attacker that is labelled **in**.

Conflict-free and admissible labellings

- Let **AF = {A, R}** be an argumentation framework and a labelling **Lab**
- **Lab** is an **admissible labelling** iff
  - every **in**-labelled argument is legally **in** and
  - every **out**-labelled argument is legally **out**.
- **Lab** is **conflict-free** iff for each argument $a \in A$ it holds that:
  - If **a** is labelled **in** then it does not have an attacker that is labelled **in**.
  - If **a** is labelled **out** then it has at least one attacker that is labelled **in**.
- Every admissible labelling is conflict-free.

Complete labellings

- **Lab** is a **complete labelling** iff
  - every **in**-labelled argument is legally **in**,
  - every **out**-labelled argument is legally **out** and
  - every **undec**-labelled argument is legally **undec**

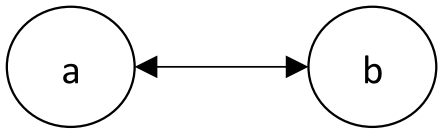
An equivalent definition

- **Lab** is a **complete labelling** iff for each argument $a \in A$ it holds that:
  - **a** is labelled **in** iff all its attackers are labelled **out**.
  - **a** is labelled **out** iff it has at least one attacker that is labelled **in**.
- Every complete labelling is admissible.

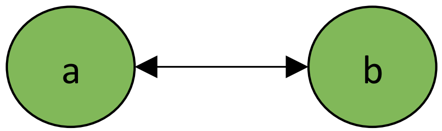
Grounded, preferred and stable labellings

- **Lab** is a **grounded labelling** iff it is complete and **in(Lab)** (the set of arguments that are labelled **in**) is minimal among the complete labellings.
- **Lab** is a **preferred labelling** iff it is complete and **in(Lab)** is maximal among the complete labellings.
- **Lab** is a **stable labelling** iff it is complete and no argument is labelled **undec**.

An example



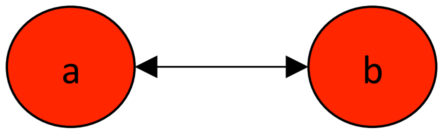
An example



Lab(a) = in
Lab(b) = in

Conflict-free	✘
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

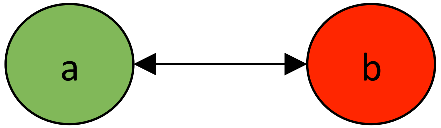
An example



Lab(a) = out
Lab(b) = out

Conflict-free	✘
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

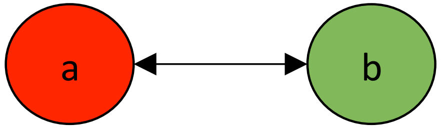
An example



Lab(a) = in
Lab(b) = out

Conflict-free	✓
Admissible	✓
Complete	✓
Grounded	✗
Preferred	✓
Stable	✓

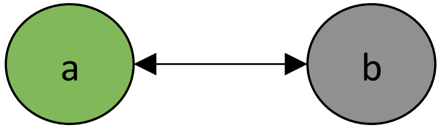
An example



Lab(a) = out
Lab(b) = in

Conflict-free	✓
Admissible	✓
Complete	✓
Grounded	✘
Preferred	✓
Stable	✓

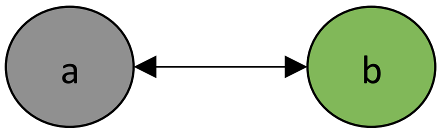
An example



Lab(a) = in
Lab(b) = undec

Conflict-free	✓
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

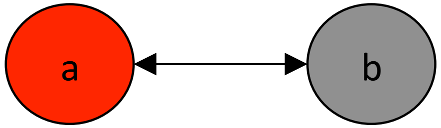
An example



Lab(a) = undec
Lab(b) = in

Conflict-free	✓
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

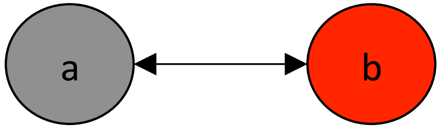
An example



Lab(a) = out
Lab(b) = undec

Conflict-free	✘
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

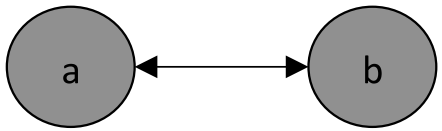
An example



Lab(a) = undec
Lab(b) = out

Conflict-free	✘
Admissible	✘
Complete	✘
Grounded	✘
Preferred	✘
Stable	✘

An example



Lab(a) = undec
Lab(b) = undec

Conflict-free	✓
Admissible	✓
Complete	✓
Grounded	✓
Preferred	✗
Stable	✗

Relation between extensions and labellings

- There is one-to-one relationship between most types of labellings and their corresponding extensions. Consider the following two functions:
- A function that takes as input a labelling Lab and returns the set of arguments that are **in**:

  Lab2Ext(Lab) = in(Lab)

- A function that take as input an extension E and returns a set of labelled arguments:

  Ext2Lab(E) = {(a,in)|a ∈ E} ∪ {(b,out)|b is attacked by E} ∪ {(c,undec)|c ∉ E and c is not attacked by E}

- Then for the complete, preferred, grounded and stable semantics it holds:
  - If **E** is a **co/pr/gr/st** extension then **Ext2Lab(E)** is a **co/pr/gr/st** labelling
  - If **Lab** is a **co/pr/gr/st** labelling then **Lab2Ext(Lab)** is a **co/pr/gr/st** extension

Summing up

- Abstract Argumentation Frameworks is a **simple** but **powerful** model of arguments and argumentation-based inference.
  - **Simple:** It treats arguments as atomic entities (without an internal structure) and uses a single binary relation to model any type of attack.
  - **Powerful:** It enables many different ways of assessing the acceptability of arguments (acceptability semantics), each implementing a different form of non-monotonic reasoning.
  - It has been shown that several non-monotonic logics (Default Logic, Defeasible Logic, Logic Programming with negation as failure, etc.) are instances of Abstract Argumentation Frameworks.