Towards a framework for the implementation and verification of translations between argumentation models
(Extended away day version)

Bas van Gijzel

University of Nottingham

June 13, 2013
Outline

1. Argumentation theory: a perceived problem

2. An introduction and implementation of argumentation frameworks (Dung)

3. Conclusions and future work
1. Argumentation theory: a perceived problem

2. An introduction and implementation of argumentation frameworks (Dung)

3. Conclusions and future work
Argumentation theory

Interdisciplinary area with various applications:
Interdisciplinary area with various applications:

- **Law:**
  Systems *modelling* legal problems/cases,
Argumentation theory

Interdisciplinary area with various applications:

- **Law:**
  Systems *modelling* legal problems/cases,

- **Decision making:**
  *Organising* information and source of *efficiency* in decision theory,
Argumentation theory

Interdisciplinary area with various applications:

- **Law:**
  Systems *modelling* legal problems/cases,

- **Decision making:**
  *Organising* information and source of *efficiency* in decision theory,

- **Communication theory:**
  *Making* argumentation in existing texts *precise.*
Argumentation theory

Interdisciplinary area with various applications:

- **Law:** Systems *modelling* legal problems/cases,
- **Decision making:** Organising information and source of *efficiency* in decision theory,
- **Communication theory:** Making argumentation in existing texts *precise*.

All these topics can give rise to *different notions of argument* and therefore *different argumentation models*. 
Argumentation theory

Interdisciplinary area with various applications:

• **Law:**
  Systems *modelling* legal problems/cases,

• **Decision making:**
  *Organising* information and source of *efficiency* in decision theory,

• **Communication theory:**
  *Making* argumentation in existing texts *precise.*

All these topics can give rise to different notions of argument and therefore different argumentation models.
(Even different notions within the topics)
Abstract argumentation

Dung’s (abstract) argumentation framework are a golden standard of argumentation.
Abstract argumentation

Dung’s (abstract) argumentation framework are a golden standard of argumentation.

- Most models are an instantiation of Dung’s model (are translatable to)
Abstract argumentation

Dung’s (abstract) argumentation framework are a golden standard of argumentation.

- Most models are an instantiation of Dung’s model (are translatable to)
- Relatively simple data structures/algorithms (complexity still NP-complete or higher for most problems)
Abstract argumentation

Dung’s (abstract) argumentation framework are a golden standard of argumentation.

• Most models are an instantiation of Dung’s model (are translatable to)
• Relatively simple data structures/algorithms (complexity still NP-complete or higher for most problems)
• Not too hard to switch between implementations of AF’s because of the very basic data structure (a directed graph)
A perceived problem

• Lack of implementations of more complex argumentation models
A perceived problem

• Lack of implementations of more complex argumentation models
• Some recent efforts to optimise the evaluation of AF’s (and ASP)
A perceived problem

- Lack of implementations of more complex argumentation models
- Some recent efforts to optimise the evaluation of AF’s (and ASP)
- Existing translations from complex models to Dung, however again a lack of implementations
A perceived problem

• Lack of implementations of more complex argumentation models
• Some recent efforts to optimise the evaluation of AF’s (and ASP)
• Existing translations from complex models to Dung, however again a lack of implementations
  • Translations are complex
  • Proofs of correctness are complex (page long proofs)
A proposed solution

• Provide implementations of Dung and some other models (Carneades, ASPIC+)
A proposed solution

• Provide implementations of Dung and some other models (Carneades, ASPIC+)
  • In a tutorial-like fashion,
  • Close to the actual mathematical definitions
A proposed solution

• Provide implementations of Dung and some other models (Carneades, ASPIC+)
  • In a tutorial-like fashion,
  • Close to the actual mathematical definitions

• In the same fashion: implement a translation
A proposed solution

- Provide implementations of Dung and some other models (Carneades, ASPIC+)
  - In a tutorial-like fashion,
  - Close to the actual mathematical definitions
- In the same fashion: implement a translation
- Provide a formalisation of implementations and translation
A proposed solution

• Provide implementations of Dung and some other models (Carneades, ASPIC+)
  • In a tutorial-like fashion,
  • Close to the actual mathematical definitions
• In the same fashion: implement a translation
• Provide a formalisation of implementations and translation

Result: a verified way to translate (unimplemented) models to an efficiently implemented model.
Outline

1. Argumentation theory: a perceived problem
2. An introduction and implementation of argumentation frameworks (Dung)
3. Conclusions and future work
Typical argument structure

Typical argument structure:

• a set of assumptions or premises,
• a method of reasoning or deduction,
• a conclusion.
Typical argument structure:

- a set of assumptions or premises,
- a method of reasoning or deduction,
- a conclusion.

\[
\begin{align*}
\text{rain} \\
\text{wet}
\end{align*}
\]
Typical argument structure:

- a set of assumptions or premises,
- a method of reasoning or deduction,
- a conclusion.

\[
\text{rain} \quad \text{wet}
\]

Note that not all models imply a strictly formal structure.
Carneades argument structures (1)

Arguments contain:
Carneades argument structures (1)

Arguments contain:

- a set of premises and exceptions
Carneades argument structures (1)

Arguments contain:

- a set of premises and exceptions
- an inference step, called applicability
Arguments contain:

• a set of premises and exceptions
• an inference step, called applicability
• another inference step called acceptability
Carneades argument structures (1)

Arguments contain:

• a set of premises and exceptions
• an inference step, called applicability
• another inference step called acceptability
• weights, used in acceptability
Carneades argument structures (2)
In 1995, Dung gave an abstract account of argumentation.

• Was able to model several contemporary approaches to non-monotonic logic,
Dung’s argumentation frameworks (AFs)

In 1995, Dung gave an abstract account of argumentation.  
- Was able to model several contemporary approaches to non-monotonic logic,
- Some scholars believe it to be too abstract,
Dung’s argumentation frameworks (AFs)

In 1995, Dung gave an *abstract* account of argumentation.

- Was able to *model* several contemporary approaches to non-monotonic logic,
- Some scholars believe it to be too *abstract*,
- However the model can be *instantiated* with more structure
Dung’s argumentation frameworks (AFs)

In 1995, Dung gave an abstract account of argumentation.

• Was able to model several contemporary approaches to non-monotonic logic,
• Some scholars believe it to be too abstract,
• However the model can be instantiated with more structure

For instance: Carneades is translatable to Dung
An abstract argumentation framework (AF) is a tuple $AF = \langle \text{Args}, \text{Def} \rangle$ such that:

• $\text{Args}$ is a set of (abstract) arguments,
• $\text{Def} \subseteq \text{Args} \times \text{Args}$.

In other words a directed graph.
An abstract argumentation framework (AF) is a tuple $AF = \langle \text{Args}, \text{Def} \rangle$ such that:

- $\text{Args}$ is a set of (abstract) arguments,
- $\text{Def} \subseteq \text{Args} \times \text{Args}$.
An abstract argumentation framework (AF) is a tuple \( AF = \langle \text{Args}, \text{Def} \rangle \) such that:

- \( \text{Args} \) is a set of (abstract) arguments,
- \( \text{Def} \subseteq \text{Args} \times \text{Args} \).

In other words a directed graph.

**Definition**
An abstract argumentation framework (AF) is a tuple $AF = \langle \text{Args}, \text{Def} \rangle$ such that:

- $\text{Args}$ is a set of (abstract) arguments,
- $\text{Def} \subseteq \text{Args} \times \text{Args}$.

In other words a directed graph.

$$A \longrightarrow B \longrightarrow C$$
Given $AF = \langle \text{Args}, \text{Def} \rangle$
Given $AF = \langle \text{Args}, \text{Def} \rangle$

Considering arguments as Strings:
AFs in Haskell

Given $AF = \langle \text{Args}, \text{Def} \rangle$

Considering arguments as Strings:

```haskell
data DungAF arg = AF [arg] [(arg, arg)]
deriving (Show)

type AbsArg = String
```
AFs in Haskell

Given $AF = \langle \text{Args}, \text{Def} \rangle$
Considering arguments as Strings:

```haskell
data DungAF arg = AF [arg] [(arg, arg)]
  deriving (Show)

type AbsArg = String
```

$$A \rightarrow B \rightarrow C$$
AFs in Haskell

Given $AF = \langle \text{Args, Def} \rangle$
Considering arguments as Strings:

```haskell
data DungAF arg = AF [arg] [(arg, arg)]
deriving (Show)
type AbsArg = String
```

And in Haskell:

```haskell
a, b, c :: AbsArg
a = "A"
b = "B"
c = "C"

AF_1 :: DungAF AbsArg
AF_1 = AF [a, b, c] [(a, b), (b, c)]
```
Attacking with a set of arguments

Given $AF = \langle \text{Args}, \text{Def} \rangle$. 
Attacking with a set of arguments

Given $AF = \langle Args, Def \rangle$.

A set $S \subseteq Args$ of arguments attacks an argument $A \in Args$...
Attacking with a set of arguments

Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments **attacks** an argument $A \in \text{Args}$ iff there exists a $B \in S$ such that $(B, A) \in \text{Def}$.
Attacking with a set of arguments

Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments attacks an argument $A \in \text{Args}$ iff there exists a $B \in S$ such that $(B, A) \in \text{Def}$.

In Haskell:

$$\text{setAttacks :: Eq arg } \Rightarrow \text{DungAF arg } \rightarrow \text{[arg] } \rightarrow \text{arg } \rightarrow \text{Bool}$$

$$\text{setAttacks (AF } \_ \text{ def) args arg}$$

$$= \text{or [b } \equiv \text{ arg | (a, b) } \leftarrow \text{ def, a } \in \text{ args]}$$
Attacking with a set of arguments

Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments attacks an argument $A \in \text{Args}$ iff there exists a $B \in S$ such that $(B, A) \in \text{Def}$.

In Haskell:

```haskell
setAttacks :: Eq arg \Rightarrow DungAF arg \to [arg] \to arg \to Bool
setAttacks (AF \_ def) args arg
  = or \[ b \equiv arg \mid (a, b) \leftarrow def, a \in args \]
```

Note that by the required $Eq \ arg \Rightarrow$, Haskell forces us to see that we need an equality on arguments to be able implement these functions.
Conflict-freeness

Given $AF = \langle \text{Args}, \text{Def} \rangle$. 
Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments is called conflict-free iff
Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments is called conflict-free iff there is no $A, B \in S$ such that $(A, B) \in \text{Def}$. 
Conflict-freeness

Given $AF = \langle \text{Args}, \text{Def} \rangle$.

A set $S \subseteq \text{Args}$ of arguments is called conflict-free iff there is no $A, B \in S$ such that $(A, B) \in \text{Def}$.

$$\text{conflictFree} :: Eq \ arg \Rightarrow \text{DungAF} \ arg \rightarrow [\ arg ] \rightarrow \text{Bool}$$

$$\text{conflictFree} \ (AF \ _\ def) \ args$$

$$= \text{null} \ [(a, b) | (a, b) \leftarrow \text{def}, a \in \text{args}, b \in \text{args}]$$
Acceptability

An argument $A \in \text{Args}$ is acceptable with respect to a set $S$ of arguments, iff for all arguments $B \in S$: if $(B, A) \in \text{Def}$ then there is a $C \in S$ for which $(C, B) \in \text{Def}$.
An argument $A \in \text{Args}$ is **acceptable** with respect to a set $S$ of arguments, iff for all arguments $B \in S$: if $(B,A) \in \text{Def}$ then there is a $C \in S$ for which $(C,B) \in \text{Def}$.

Alternatively $S$ **defends** $A$, 

**Acceptability**
Acceptability

An argument $A \in \text{Args}$ is acceptable with respect to a set $S$ of arguments, iff for all arguments $B \in S$: if $(B, A) \in \text{Def}$ then there is a $C \in S$ for which $(C, B) \in \text{Def}$.

Alternatively $S$ defends $A$,

$$
\text{acceptable} :: \text{Eq arg} \Rightarrow \text{DungAF arg} \rightarrow \text{arg} \rightarrow \text{[arg]} \rightarrow \text{Bool}
$$

$$
\text{acceptable af}(\text{AF} \_ \text{def}) \_ \text{a} \_ \text{args}
= \text{and} \ [\text{setAttacks af args } b \mid (b, a') \leftarrow \text{def}, a \equiv a']
$$
The characteristic function of an AF, $F_{AF} : 2^{\text{Args}} \rightarrow 2^{\text{Args}}$, is a function,
The characteristic function of an AF, $F_{AF} : 2^{\text{Args}} \to 2^{\text{Args}}$, is a function, such that, given a set of arguments $S$, $F_{AF}(S) = \{A \mid A$ is acceptable w.r.t. to $S\}$. 
The characteristic function of an $AF$, $F_{AF} : 2^{\text{Args}} \rightarrow 2^{\text{Args}}$, is a function, such that, given a set of arguments $S$, $F_{AF}(S) = \{A \mid A$ is acceptable w.r.t. to $S\}$.

Given that $F_{AF}$ is ordered by the subset relation and $S$ is conflict-free, then $F_{AF}$ is monotonic.
The characteristic function of an AF, $F_{AF} : 2^{\text{Args}} \rightarrow 2^{\text{Args}}$, is a function, such that, given a set of arguments $S$, $F_{AF}(S) = \{ A \mid A \text{ is acceptable w.r.t. to } S \}$.

Given that $F_{AF}$ is ordered by the subset relation and $S$ is conflict-free, then $F_{AF}$ is monotonic.

$$f :: \text{Eq arg} \Rightarrow \text{DungAF arg} \rightarrow [\text{arg}] \rightarrow [\text{arg}]$$

$$f \ af@(AF \ args _) \ s = [a \mid a \leftarrow \text{args}, \text{acceptable af a s}]$$
An extension is a

“set of arguments that are acceptable when taken together”
An extension is a

“set of arguments that are acceptable when taken together”

The grounded extension is the minimally acceptable set.
Grounded extension (2)

Given a conflict-free set of arguments $S$ and argumentation framework $AF$: 
Grounded extension (2)

Given a conflict-free set of arguments $S$ and argumentation framework $AF$:

$S$ is a grounded extension iff it is the least fixed point of $F_{AF}$. 
Grounded extension in Haskell

*S* is a grounded extension iff it is the least fixed point of $F_{AF}$.

$$AF_1 :: DungAF \ AbsArg$$

$$AF_1 = AF \ [a, b, c] \ [(a, b), (b, c)]$$

$$f_{AF_1} :: [AbsArg] \rightarrow [AbsArg]$$

$$f_{AF_1} = f \ AF_1$$
Grounded extension in Haskell

$S$ is a grounded extension iff it is the least fixed point of $F_{AF}$.

$$AF_1 :: DungAF \ AbsArg$$
$$AF_1 = AF [a, b, c] [(a, b), (b, c)]$$

$$f_{AF_1} :: [AbsArg] \rightarrow [AbsArg]$$

$$f_{AF_1} = f \ AF_1$$

$$groundedF :: Eq \ arg \Rightarrow ([arg] \rightarrow [arg]) \rightarrow [arg]$$

$$groundedF \ f = groundedF' \ f \ []$$

**where**

$$groundedF' \ f \ args$$

$$| f \ args \equiv args = args$$

$$| otherwise = groundedF' \ f \ (f \ args)$$
Grounded extension in Haskell

S is a grounded extension iff it is the least fixed point of \( F_{AF} \).

\[
AF_1 :: \text{DungAF AbsArg}
\]
\[
AF_1 = AF [a, b, c] [(a, b), (b, c)]
\]
\[
f_{AF_1} :: [\text{AbsArg}] \rightarrow [\text{AbsArg}]
\]
\[
f_{AF_1} = f \ AF_1
\]

\[
groundedF :: \text{Eq arg} \Rightarrow ([\text{arg}] \rightarrow [\text{arg}]) \rightarrow [\text{arg}]
\]
\[
groundedF f = \text{groundedF}' f []
\]

where \( \text{groundedF}' f \) args
\[
\mid f \ args \equiv \text{args} = \text{args}
\]
\[
\mid \text{otherwise} \quad = \text{groundedF}' f (f \ args)
\]

Then as expected:

\[
groundedF \ f_{AF_1} > ["A", "C"]
\]
Outline

1. Argumentation theory: a perceived problem
2. An introduction and implementation of argumentation frameworks (Dung)
3. Conclusions and future work
Overview of work done (1)

• Large parts of Dung’s definition have been implemented in Haskell,
Overview of work done (1)

• Large parts of Dung’s definition have been implemented in Haskell,
• Most of these definitions have been formalised in Agda,
Overview of work done (1)

• Large parts of Dung’s definition have been implemented in Haskell,
• Most of these definitions have been formalised in Agda,
• In previous work we implemented Carneades in Haskell,
Overview of work done (1)

• Large parts of Dung’s definition have been implemented in Haskell,
• Most of these definitions have been formalised in Agda,
• In previous work we implemented Carneades in Haskell,
• Provided a sketch of how to do a translation from Carneades to Dung in Haskell and which properties one would want to prove.
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
- (Almost) Cabalised and uploaded the Dung implementation to Hackage,
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
- (Almost) Cabalised and uploaded the Dung implementation to Hackage,
- Cabalised and uploaded the Carneades implementation to Hackage,
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
- (Almost) Cabalised and uploaded the Dung implementation to Hackage,
- Cabalised and uploaded the Carneades implementation to Hackage,
- Installation instructions (hopefully) usable for argumentation theorists.
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
- (Almost) Cabalised and uploaded the Dung implementation to Hackage,
- Cabalised and uploaded the Carneades implementation to Hackage,
- Installation instructions (hopefully) usable for argumentation theorists.

This has caused some people to pick this up (used as a course in Edinburgh by Alan Smaill).
Overview of work done (2)

- All code is or will be available as literate Haskell/Agda,
- (Almost) Cabalised and uploaded the Dung implementation to Hackage,
- Cabalised and uploaded the Carneades implementation to Hackage,
- Installation instructions (hopefully) usable for argumentation theorists.

This has caused some people to pick this up (used as a course in Edinburgh by Alan Smaill).

Formalisation in Agda, the initial work on the translation and all Haskell code is either discussed or linked to in the paper.
Conclusion

- High-level Haskell code close to the mathematical definitions:
Conclusion

• High-level Haskell code close to the mathematical definitions:
  • Allowing greater understanding of the implementation,
Conclusion

• High-level Haskell code close to the **mathematical definitions:**
  • Allowing **greater understanding** of the implementation,
  • Written in a notation **closely related** to that of argumentation theorists.
Conclusion

• High-level Haskell code close to the mathematical definitions:
  • Allowing greater understanding of the implementation,
  • Written in a notation closely related to that of argumentation theorists.
• Agda formalisation of the Dung implementation:
Conclusion

• High-level Haskell code close to the mathematical definitions:
  • Allowing greater understanding of the implementation,
  • Written in a notation closely related to that of argumentation theorists.

• Agda formalisation of the Dung implementation:
  • The first formalisation (to my knowledge) of an argumentation model,
Conclusion

• High-level Haskell code close to the mathematical definitions:
  • Allowing greater understanding of the implementation,
  • Written in a notation closely related to that of argumentation theorists.

• Agda formalisation of the Dung implementation:
  • The first formalisation (to my knowledge) of an argumentation model,
  • Easier realisation and formalisation of existing/future translations,
Conclusion

• High-level Haskell code close to the mathematical definitions:
  • Allowing greater understanding of the implementation,
  • Written in a notation closely related to that of argumentation theorists.

• Agda formalisation of the Dung implementation:
  • The first formalisation (to my knowledge) of an argumentation model,
  • Easier realisation and formalisation of existing/future translations,
  • A better understanding of the meaning of some of the complexer argumentation models.
Future work

• Further formalisation of Dung’s definition and theorems:
Future work

- **Further formalisation** of Dung’s definition and theorems:
  - Formalisation of fixpoints in Agda is a lot of work!
Future work

• Further formalisation of Dung’s definition and theorems:
  • Formalisation of fixpoints in Agda is a lot of work!
• Implementation and formalisation of the translation from Carneades to Dung.
Future work

• **Further formalisation** of Dung’s definition and theorems:
  • Formalisation of fixpoints in Agda is a lot of work!
• **Implementation and formalisation** of the translation from Carneades to Dung.
  • Will involve doing some formal work to **refactor** out the intermediate translation to ASPIC⁺,
Future work

- **Further formalisation** of Dung’s definition and theorems:
  - Formalisation of fixpoints in Agda is a lot of work!
- **Implementation and formalisation** of the translation from Carneades to Dung.
  - Will involve doing some formal work to refactor out the intermediate translation to ASPIC$^+$,
  - Might switch to Coq if Agda becomes infeasible.
Future work

• **Further formalisation** of Dung’s definition and theorems:
  • Formalisation of fixpoints in Agda is a lot of work!
• **Implementation and formalisation** of the translation from Carneades to Dung.
  • Will involve doing some formal work to refactor out the intermediate translation to ASPIC\(^+\),
  • Might switch to Coq if Agda becomes infeasible.
• **Implement and translate(?)** my generalisation of the ASPIC\(^+\) argumentation model