Tools for the implementation of argumentation models

Bas van Gijzel

Functional Programming Laboratory
University of Nottingham

September 26, 2013
Outline

1 Argumentation theory: a perceived problem
2 Demo time
3 Conclusions
Outline

1. Argumentation theory: a perceived problem
2. Demo time
3. Conclusions
Argumentation theory

Interdisciplinary area with various applications:
Argumentation theory

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/linguistics:**
  *Making* argumentation in existing texts *precise*.
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
• One or more relations on this set, such as attack
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
• One or more relations on this set, such as attack

Structured:

• A defined structure of arguments:
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
• One or more relations on this set, such as attack

Structured:

• A defined structure of arguments:
  • Knowledge base and rules,
  • Argument schemes,
  • etc.
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
• One or more relations on this set, such as attack

Structured:

• A defined structure of arguments:
  • Knowledge base and rules,
  • Argument schemes,
  • etc.
• Additional mathematical structures:
Mathematical models of argumentation

Formal argumentation theory deals with mathematical models of argumentation.

Abstract:

• No defined structure of arguments, just a set
• One or more relations on this set, such as attack

Structured:

• A defined structure of arguments:
  • Knowledge base and rules,
  • Argument schemes,
  • etc.
• Additional mathematical structures:
  • Weights,
  • Audiences,
  • Agents,
  • etc.
Implementations of abstract models of argumentations

Dung’s (abstract) argumentation frameworks are most commonly used, and like a gold standard of argumentation.
Implementations of abstract models of argumentations

Dung’s (abstract) argumentation frameworks are most commonly used, and like a gold standard of argumentation.

- Formally related to logic programming,
Implementations of abstract models of argumentations

Dung’s (abstract) argumentation frameworks are most commonly used, and like a gold standard of argumentation.

• Formally related to logic programming,
• Therefore, intuitively implementable in Prolog,
Implementations of abstract models of argumentations

Dung’s (abstract) argumentation frameworks are most commonly used, and like a gold standard of argumentation.

- Formally related to logic programming,
- Therefore, intuitively implementable in Prolog,
- Efficiently (and intuitively) implementable in Answer Set Programming
Implementations of structured models of argumentations?

• Lack of (documented) implementations of more complex argumentation models
• Most are unavailable or closed source

Existing translations from structured models to Dung:
• Translations are complex
• Proofs of correctness are complex (page long proofs)
• Barely any existing implementation

Structured argumentation models need their logic programming/ASP!
Implementations of structured models of argumentations?

Not so much:

• Lack of (documented) implementations of more complex argumentation models
Implementations of structured models of argumentations?

Not so much:

• Lack of (documented) implementations of more complex argumentation models
• Most are unavailable or closed source
Implementations of structured models of argumentations?

Not so much:

• Lack of (documented) implementations of more complex argumentation models

• Most are unavailable or closed source

Existing translations from structured models to Dung, however:
Implementations of structured models of argumentations?

Not so much:

• Lack of (documented) implementations of more complex argumentation models
• Most are unavailable or closed source

Existing translations from structured models to Dung, however:

• Translations are complex
• Proofs of correctness are complex (page long proofs)
• Barely any existing implementation
Implementations of structured models of argumentations?

Not so much:

• Lack of (documented) implementations of more complex argumentation models
• Most are unavailable or closed source

Existing translations from structured models to Dung, however:

• Translations are complex
• Proofs of correctness are complex (page long proofs)
• Barely any existing implementation

Structured argumentation models need their logic programming/ASP!
Methods and tools to the rescue!

Functional programming:

• Again, close to the actual mathematics,
• Allows high-level code, both for functions and for structure.

Tools:

• Literate programming:
  • Intermixing of documentation, definition and implementation,
  • Machine-readable.

• Open source + repositories:
  • All (literate) programming code is public,
  • Almost instantly installable through package repositories.
Methods and tools to the rescue!

Functional programming:

- Again, close to the actual mathematics,
Methods and tools to the rescue!

Functional programming:

• Again, close to the actual mathematics,
• Allows high-level code, both for functions and for structure.
Methods and tools to the rescue!

Functional programming:

• Again, close to the actual mathematics,
• Allows high-level code, both for functions and for structure.

Tools:

• Literate programming:
  • Intermixing of documentation, definition and implementation,
Methods and tools to the rescue!

Functional programming:

- Again, close to the actual mathematics,
- Allows high-level code, both for functions and for structure.

Tools:

- Literate programming:
  - Intermixing of documentation, definition and implementation,
  - Machine-readable.
Methods and tools to the rescue!

Functional programming:
• Again, close to the actual mathematics,
• Allows high-level code, both for functions and for structure.

Tools:
• Literate programming:
  • Intermixing of documentation, definition and implementation,
  • Machine-readable.
• Open source + repositories:
Methods and tools to the rescue!

Functional programming:
  • Again, close to the actual mathematics,
  • Allows high-level code, both for functions and for structure.

Tools:
  • Literate programming:
    • Intermixing of documentation, definition and implementation,
    • Machine-readable.
  • Open source + repositories:
    • All (literate) programming code is public,
Methods and tools to the rescue!

Functional programming:

• Again, close to the actual mathematics,
• Allows high-level code, both for functions and for structure.

Tools:

• Literate programming:
  • Intermixing of documentation, definition and implementation,
  • Machine-readable.
• Open source + repositories:
  • All (literate) programming code is public,
  • Almost instantly installable through package repositories.
Outline

1 Argumentation theory: a perceived problem

2 Demo time

3 Conclusions
Outline

1. Argumentation theory: a perceived problem
2. Demo time
3. Conclusions
Overview of work done (1)

- Carneades has been **fully implemented in Haskell**, 
Overview of work done (1)

- Carneades has been fully implemented in Haskell,
- Large parts of Dung’s definitions have been implemented in Haskell,
Overview of work done (1)

- Carneades has been fully implemented in Haskell,
- Large parts of Dung’s definitions have been implemented in Haskell,
- All code is available as literate Haskell,
Overview of work done (1)

- Carneades has been fully implemented in Haskell,
- Large parts of Dung’s definitions have been implemented in Haskell,
- All code is available as literate Haskell,
  - Paper (and if needed, even the slides) can be loaded into the compiler
Overview of work done (2)

- Cabalised and uploaded the Carneades implementation to Hackage,
Overview of work done (2)

- Cabalised and uploaded the Carneades implementation to Hackage,
- Cabalised and uploaded the Dung implementation to Hackage,
Overview of work done (2)

• **Cabalised and uploaded** the Carneades implementation to Hackage,

• **Cabalised and uploaded** the Dung implementation to Hackage,

• **Installation instructions** (hopefully) usable for non-experienced programmers.
Overview of work done (2)

• **Cabalised and uploaded** the Carneades implementation to Hackage,
• **Cabalised and uploaded** the Dung implementation to Hackage,
• **Installation instructions** (hopefully) usable for non-experienced programmers.

This has caused some people to pick this up:
Overview of work done (2)

• Cabalised and uploaded the Carneades implementation to Hackage,
• Cabalised and uploaded the Dung implementation to Hackage,
• Installation instructions (hopefully) usable for non-experienced programmers.

This has caused some people to pick this up:

• Carneades package used in an AI module in Edinburgh by Alan Smaill
Overview of work done (2)

- Cabalised and uploaded the Carneades implementation to Hackage,
- Cabalised and uploaded the Dung implementation to Hackage,
- Installation instructions (hopefully) usable for non-experienced programmers.

This has caused some people to pick this up:

- Carneades package used in an AI module in Edinburgh by Alan Smaill
- Dung package used in an NLP library by Jann Müller
Future work

Verification of implementations!
Future work

Verification of implementations!

Done:

- Formalisation of Dung’s definitions in a theorem prover:

Todo:

- Formalisation of Carneades’ definitions in a theorem prover
- Formalisation of the translation from Carneades to Dung
- Connect the implementation of Dung’s AFs to an optimised implementation using ASP or SAT
Future work

Verification of implementations!

Done:

• Formalisation of Dung’s definitions in a theorem prover:
• Implementation of the translation from Carneades to Dung
Future work

Verification of implementations!

Done:
• **Formalisation** of Dung’s definitions in a theorem prover:
• **Implementation** of the translation from Carneades to Dung

Todo:
• **Formalisation** of Carneades’ definitions in a theorem prover
Future work

Verification of implementations!

Done:

• **Formalisation** of Dung’s definitions in a theorem prover:
• **Implementation** of the translation from Carneades to Dung

Todo:

• **Formalisation** of Carneades’ definitions in a theorem prover
• **Formalisation** of the translation from Carneades to Dung.
Future work

Verification of implementations!

Done:

• Formalisation of Dung’s definitions in a theorem prover:
  • Implementation of the translation from Carneades to Dung

Todo:

• Formalisation of Carneades’ definitions in a theorem prover
• Formalisation of the translation from Carneades to Dung.
• Connect the implementation of Dung’s AFs to an optimised implementation using ASP or SAT