A Distributed Argumentation Framework using Defeasible Logic Programming

Matthias Thimm   Gabriele Kern-Isberner

Technische Universität Dortmund

May 29, 2008

Outline

Outline

Overview

- DeLP (*Defeasible Logic Programming*) consists of facts, strict and defeasible rules

$Bird(tweety).$	(fact)
$Bird(X) \leftarrow Penguin(X).$	(strict rule)
$Flies(X) \prec Bird(X).$	(defeasible rule)

- A defeasible logic program (*de.l.p.*) $\mathcal{P}$ is a tuple $\mathcal{P} = (\Pi, \Delta)$ with a set Π of facts and strict rules and a set Δ of defeasible rules.

Arguments and counterarguments

Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.*

Definition (Argument, subargument)

$\langle \mathcal{A}, h \rangle$ with $\mathcal{A} \subseteq \Delta$ is an *argument* iff

- $\mathcal{A} \cup \Pi \mathrel{|\!\sim} h$
- $\mathcal{A} \cup \Pi \mathrel{|\!\not\sim} \bot$
- $\mathcal{A}$ is minimal

$\langle \mathcal{A}_1, h_1 \rangle$ is a *subargument* of $\langle \mathcal{A}_2, h_2 \rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$.

Arguments and counterarguments

Let $\mathcal{P} = (\Pi, \Delta)$ be a *de.l.p.*

Definition (Argument, subargument)

$\langle \mathcal{A}, h \rangle$ with $\mathcal{A} \subseteq \Delta$ is an *argument* iff

- $\mathcal{A} \cup \Pi \mid\!\sim h$
- $\mathcal{A} \cup \Pi \not\mid\!\sim \bot$
- $\mathcal{A}$ is minimal

$\langle \mathcal{A}_1, h_1 \rangle$ is a *subargument* of $\langle \mathcal{A}_2, h_2 \rangle$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$.

Definition (Counterargument)

$\langle \mathcal{A}_1, h_1 \rangle$ is a *counterargument* of $\langle \mathcal{A}_2, h_2 \rangle$ at a literal h iff

$$\exists \langle \mathcal{A}, h \rangle : \mathcal{A} \subseteq \mathcal{A}_2 : \Pi \cup \{h, h_1\} \mid\!\sim \bot \quad (h \text{ and } h_1 \textit{ disagree})$$

Acceptable argumentation lines

Let $\mathcal{P}$ be a *de.l.p.*

Definition (Acceptable argumentation line)

$\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ is an *acceptable argumentation line* iff

1. Λ is finite

Acceptable argumentation lines

Let $\mathcal{P}$ be a *de.l.p.*

Definition (Acceptable argumentation line)

$\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ is an *acceptable argumentation line* iff

1. Λ is finite,
2. every argument is an attack on its predecessor;

Acceptable argumentation lines

Let $\mathcal{P}$ be a *de.l.p.*

Definition (Acceptable argumentation line)

$\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ is an *acceptable argumentation line* iff

1. Λ is finite,
2. every argument is an attack on its predecessor; there are no two consecutive blocking attacks (given a preference relation under arguments)

Acceptable argumentation lines

Let $\mathcal{P}$ be a *de.l.p.*

Definition (Acceptable argumentation line)

$\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ is an *acceptable argumentation line* iff

1. Λ is finite,
2. every argument is an attack on its predecessor; there are no two consecutive blocking attacks (given a preference relation under arguments),
3. the set of supporting arguments is consistent with respect to Π,
4. the set of interfering arguments is consistent with respect to Π

Acceptable argumentation lines

Let $\mathcal{P}$ be a *de.l.p.*

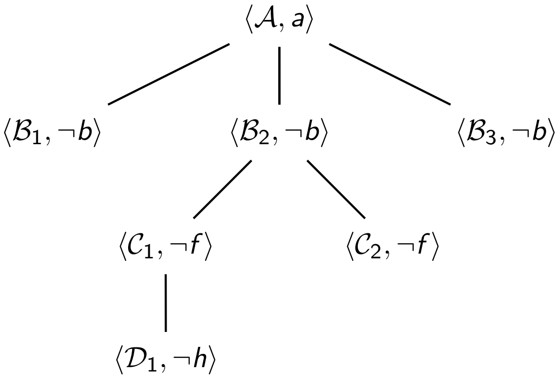
Definition (Acceptable argumentation line)

$\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ is an *acceptable argumentation line* iff

1. Λ is finite,
2. every argument is an attack on its predecessor; there are no two consecutive blocking attacks (given a preference relation under arguments),
3. the set of supporting arguments is consistent with respect to Π,
4. the set of interfering arguments is consistent with respect to Π,
5. no argument $\langle \mathcal{A}_k, h_k \rangle$ is a subargument of a preceding argument.

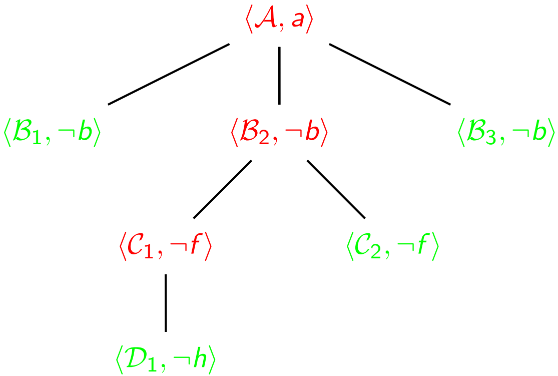
The warrant procedure

Representation of the dialectical process in a *dialectical tree*:



The warrant procedure

Representation of the dialectical process in a *dialectical tree*:



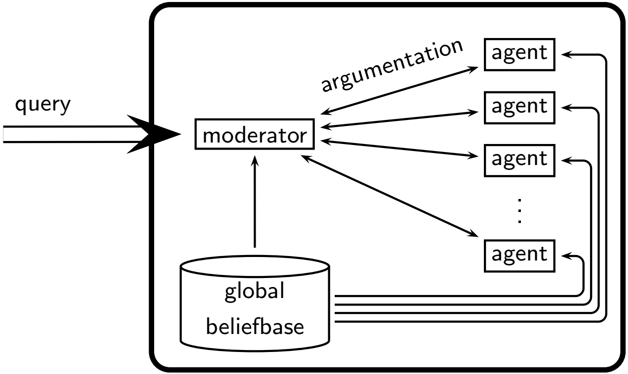
Warrant

Definition (Warrant)

A literal h is *warranted*, iff there exists an argument $\langle \mathcal{A}, h \rangle$ for h, such that the root of the marked dialectical tree $\mathcal{T}^*_{\langle \mathcal{A}, h \rangle}$ is marked "undefeated". Then $\langle \mathcal{A}, h \rangle$ is a *warrant* for h.

Outline

Overview 1/2


query
moderator
argumentation
agent
agent
agent
agent
global
beliefbase

Overview 2/2

Definition (Global belief base)

A *global belief base* Π is a non-contradictory set of facts and strict rules.

$\longrightarrow$ The global belief base consists of common beliefs.

Overview 2/2

Definition (Global belief base)

A *global belief base* Π is a non-contradictory set of facts and strict rules.

$\longrightarrow$ The global belief base consists of common beliefs.

Definition (Local belief base)

Let Δ be a set of defeasible rules and Π a global belief base. If $\Delta \cup \Pi$ is consistent (treating defeasible rules as strict rules), Δ is called *local beliefbase* relative to Π.

$\longrightarrow$ A local belief base reflects an agent's own beliefs besides the common beliefs.

The Moderator 1/2

The moderator is the central component in the architecture. He receives queries from outside, coordinates the argumentation process and returns answers.

The Moderator 1/2

The moderator is the central component in the architecture. He receives queries from outside, coordinates the argumentation process and returns answers.

Definition (Moderator)

A *moderator* is a tuple (μ, χ, η) with

- a decision function μ (evaluates a set of dialectical trees regarding a given query),

The Moderator 1/2

The moderator is the central component in the architecture. He receives queries from outside, coordinates the argumentation process and returns answers.

Definition (Moderator)

A *moderator* is a tuple (μ, χ, η) with

- a decision function μ (evaluates a set of dialectical trees regarding a given query),
- an analysis function χ (evaluates the marking of the root argument of a given dialectical tree)

The Moderator 1/2

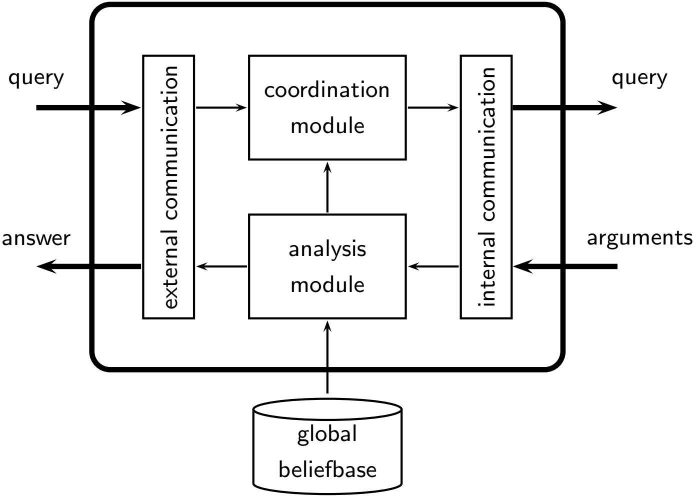
The moderator is the central component in the architecture. He receives queries from outside, coordinates the argumentation process and returns answers.

Definition (Moderator)

A *moderator* is a tuple (μ, χ, η) with

- a decision function μ (evaluates a set of dialectical trees regarding a given query),
- an analysis function χ (evaluates the marking of the root argument of a given dialectical tree) and
- an acceptance function η (checks whether a given argumentation line acceptable).

The Moderator 2/2



Agents 1/2

An agent generates initial arguments for a given literal and counterarguments for a given argument.

Agents 1/2

An agent generates initial arguments for a given literal and counterarguments for a given argument.

Definition (Agent)

An agent is a tuple $(\Delta, \varphi, \psi, \eta)$ with

- a local belief base Δ,
- a root argument function φ (returns all arguments from Δ for a given literal),

Agents 1/2

An agent generates initial arguments for a given literal and counterarguments for a given argument.

Definition (Agent)

An agent is a tuple $(\Delta, \varphi, \psi, \eta)$ with

- a local belief base Δ,
- a root argument function φ (returns all arguments from Δ for a given literal),
- a counterargument function ψ (returns all counterargument for a given argument)

Agents 1/2

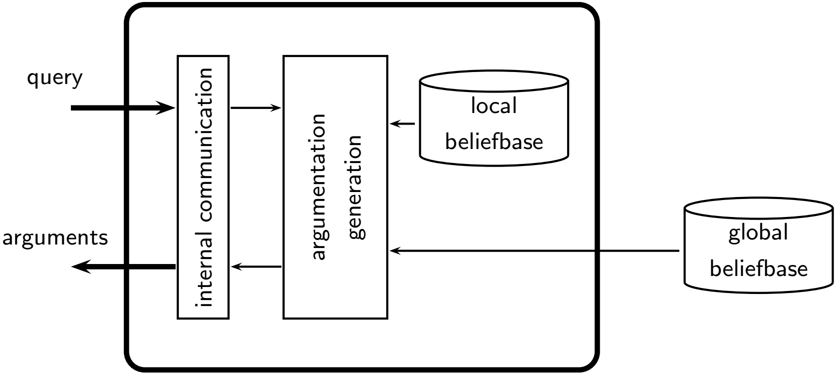
An agent generates initial arguments for a given literal and counterarguments for a given argument.

Definition (Agent)

An agent is a tuple $(\Delta, \varphi, \psi, \eta)$ with

- a local belief base Δ,
- a root argument function φ (returns all arguments from Δ for a given literal),
- a counterargument function ψ (returns all counterargument for a given argument)
- and an acceptance function η (checks whether a given argumentation line acceptable).

Agents 2/2


query
arguments
internal communication
argumentation generation
local beliefbase
global beliefbase

The argumentation process

Definition (Argumentation-based multi agent system (ArgMAS))

An *ArgMAS* is a tuple $(M, \Pi, \{A_1, \ldots, A_n\})$ with a moderator M, a global belief base Π and agents $A_1, \ldots, A_n$.

The argumentation process

Definition (Argumentation-based multi agent system (ArgMAS))

An *ArgMAS* is a tuple $(M, \Pi, \{A_1, \ldots, A_n\})$ with a moderator M, a global belief base Π and agents $A_1, \ldots, A_n$.

Definition (Argumentation product)

Let h be a query (a literal) and T an ArgMAS. An *argumentation product* υ of T anf h is a dialectical tree with:

1. The root argument of υ is an element of $\varphi_j(h)$ for a $j \in \{1, \ldots, n\}$

The argumentation process

Definition (Argumentation-based multi agent system (ArgMAS))

An *ArgMAS* is a tuple $(M, \Pi, \{A_1, \ldots, A_n\})$ with a moderator M, a global belief base Π and agents $A_1, \ldots, A_n$.

Definition (Argumentation product)

Let h be a query (a literal) and T an ArgMAS. An *argumentation product* υ of T anf h is a dialectical tree with:

1. The root argument of υ is an element of $\varphi_j(h)$ for a $j \in \{1, \ldots, n\}$
2. For every path $\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ in υ it holds for the set K of all children of $\langle \mathcal{A}_n, h_n \rangle$

$$K = \{\langle \mathcal{B}, h' \rangle | \langle \mathcal{B}, h' \rangle \in \psi_1(\Lambda) \cup \cdots \cup \psi_n(\Lambda) \wedge \eta(\Lambda + \langle \mathcal{B}, h' \rangle) = 1\}.$$

An application scenario

- Assume two agents, acting as accuser and defender in a legal dispute.
- Then the moderator can be identified with the judge.
- A reasonable query for this multi agent system would be the question of guilt of the accused.
- As a first step to answer this query, the judge asks the accuser and the defender to propose initial arguments for and against the statement "The accused is guilty".
- Both, the defender and the accuser, can react to the arguments of their counterpart with counterarguments.
- Eventually, the judge analyses the resulting argumentation lines and returns "guilty" or "not guilty" to the questioner, i. e the people.

Outline

Remarks and conclusion

- We presented a distributed and centralized approach for defeasible argumentation, which is useful in dispute scenarios.
- The proposed system was applied to a real world legal dispute and turned out well.

Remarks and conclusion

- We presented a distributed and centralized approach for defeasible argumentation, which is useful in dispute scenarios.
- The proposed system was applied to a real world legal dispute and turned out well.

- It can be shown that every *de.l.p.* can be translated into the proposed framework while preserving answer behaviour.
- There are instances of the distributed framework which can not be modeled in general DeLP.
- We are currently investigating these relationships more closely.

Remarks and conclusion

- We presented a distributed and centralized approach for defeasible argumentation, which is useful in dispute scenarios.
- The proposed system was applied to a real world legal dispute and turned out well.

- It can be shown that every *de.l.p.* can be translated into the proposed framework while preserving answer behaviour.
- There are instances of the distributed framework which can not be modeled in general DeLP.
- We are currently investigating these relationships more closely.

Thank you for your attention