Computational Models for Argumentation in MAS

Carlos Iván Chesñevar
Dept. of Computer Science
UNIVERSITAT DE LLEIDA
SPAIN

Guillermo R. Simari
Dept. of Computer Science and Engineering
UNIVERSIDAD NACIONAL DEL SUR
ARGENTINA

Where are we from...

Univ. Nacional del Sur (Bahía Blanca, Argentina)

University of Lleida (Lleida, Catalonia, Spain)
Main references


Outline

• *(Very brief)* Introduction to Multiagent Systems

• What is argumentation? Fundamentals

• A Case Study: DeLP and its extensions as an argument-based approach to logic programming.

• Argumentation meets agents: argument-based negotiation

• Conclusions
Overview

Five ongoing trends have marked the history of computing:

- **ubiquity**;
- **interconnection**;
- **intelligence**;
- **delegation**; and
- **human-orientation**

Credits: some of these slides are based on Michael Wooldridge’s lecture notes for his book “An Introduction to MAS” (Wiley & Sons, 2002)

Ubiquity, Interconnection, Intelligence

- As processing capability spreads, sophistication (and intelligence of a sort) becomes ubiquitous.
- What could benefit from having a processor embedded in it...?
- Internet is powerful...Some researchers are putting forward theoretical models that portray computing as primarily a **process of interaction**.
- The complexity of tasks that we are capable of automating and delegating to computers has grown steadily.
Computational Models for Argumentation in Multiagent Systems – EASSS 2005

Delegation, Human-Orientation

- Computers are doing more for us – without our intervention. Next on the agenda: fly-by-wire cars, intelligent braking systems…
- Programmers conceptualize and implement software in terms of higher-level – more human-oriented – abstractions.
- The movement away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way we ourselves understand the world.

Programming progression…

- Programming has progressed through:
  - machine code;
  - assembly language;
  - machine-independent programming languages;
  - sub-routines;
  - procedures & functions;
  - abstract data types;
  - objects;
- to agents.
Where does it bring us?

- Delegation and Intelligence imply the need to build computer systems that can act effectively on our behalf.

- This implies:
  - The ability of computer systems to act *independently*.
  - The ability of computer systems to act in a way that *represents our best interests* while interacting with other humans or systems.

Interconnection and Distribution

- Interconnection and Distribution have become core motifs in Computer Science.

- But Interconnection and Distribution, coupled with the need for systems to represent our best interests, implies systems that can *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people).
So Computer Science expands…

- These issues were not studied in Computer Science until recently.
- All of these trends have led to the emergence of a new field in Computer Science: Multiagent Systems.
- An agent is a computer system that is capable of independent action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told).

Multiagent Systems: a Definition

- A multiagent system is one that consists of a number of agents, which interact with one another.
- In the most general case, agents will be acting on behalf of users with different goals and motivations.
- To successfully interact, they will require the ability to cooperate, coordinate, and negotiate with each other, much as people do.
In Multiagent Systems, we address questions such as:

- How can cooperation emerge in societies of self-interested agents?
- What kinds of languages can agents use to communicate?
- How can self-interested agents recognize conflict, and how can they (nevertheless) reach agreement?
- How can autonomous agents coordinate their activities so as to cooperatively achieve goals?
Generic Agent

Sensors receive perceptions

¿What to do?

Effectors execute those chosen actions to be carried out...

Effectors


Sensors

Beliefs

Plans

Desires

Intentions

Argumentation!

Actuator

Actuator
Architecture

Sensors

Beliefs

Desires

Intentions

Plans

Actuator

Argumentation-Based Reasoning Engine!

Simple Reactive Agent

Agent

Sensors

Observations about the world

Rules Condition-action

What to do?

Effectors

Argumentation-Based Reasoning Engine!
Computational Models for Argumentation in Multiagent Systems – EASSS 2005

Reactive Agent with Inner State

Agent

State

Observations
About the world

Consequences of the actions

Rules
Condition-action

What to do?

Effectors

Agent

Environment

Sensors

Observations
About the current State of The world

Which are the conseq.
of doing action A

What to do now?

Effectores

Agent with Explicit Goals

Agent

State

Observations
About the current State of The world

Which are the conseq.
of doing action A

What to do now?

Effectores

Agent

Environment

Sensors

Agent with Explicit Goals

Agent

State

Consequences from Actions

Goals

Reactive Agent with Inner State

Argumentation-Based Reasoning Engine!

Argumentation-Based Reasoning Engine!
Utility-based agent

Agent

State

Observations
About the current state
Of The world

Consequences from
actions

Which are the
consequences
Of doing action A?

How good is the state
I would achieve?

What to do now?

Effectors

Outline

• (Very brief) Introduction to Multiagent Systems

• What is argumentation? Fundamentals

• A Case Study: DeLP and its extensions as an argument-based approach to logic programming.

• Argumentation meets agents: argument-based negotiation

• Conclusions
Systems for defeasible argumentation. Generalities

Typical problems in (non-monotonic) default reasoning:

1) Representation of defaults: e.g. Birds usually fly
2) Inconsistency handling: identify relevant subsets of consistent information.
3) Identifying preferred models

Many approaches have been developed:

- Default logic (Reiter, 1980)
- Preferred subtheories (Brewka, 1989)
- Circumscription (McCarthy, 1987)
- Others...

Argumentation systems (AS) are “yet another way” to formalize common-sense reasoning. Non-monotonicity arises from the fact that new premises may give rise to stronger counterarguments, which in turn will defeat the original argument.

1) Normality condition view: an argument = standard proof from a set of premises + normality statements. A counterargument is an attack on such a normality statement.
2) Inconsistency handling view: an argument = standard proof from a consistent subset of the premises. A counterargument is an attack on a premise of an argument.
3) Semantic view: constructing ‘invalid’ arguments (wrt the semantics) is allowed in the proof theory. A counterargument is an attack on the use of an inference rule which deviates from a preferred model.
According to Prakken & Vreeswijk (2002), there are five common elements to systems for defeasible argumentation:

- Definition of Underlying Logical Language
- Definition of Argument
- Definition of Conflict among Arguments
- Definition of Defeat among Arguments
- Definition of Status of Arguments

The underlying logic: Arguments & Logical consequence

- Argumentation Systems are constructed starting from a **logical language** and an associated notion of **logical consequence** for that language.
- The logical consequence relation helps to define what will be considered an **argument**.
- This consequence relation is **monotonic**, i.e., new information cannot invalidate arguments as such, but rather give rise to counterarguments.
- Arguments are seen as **proofs** in the chosen logic.
Argument as a ‘proof’

Arguments are presented under different forms:

- An inference tree grounded in premises.
- A deduction sequence.
- A pair \((\text{Premises}, \text{Conclusion})\), leaving unspecified the particular proof, in the underlying logic, that leads from the \text{Premises} to the \text{Conclusion}.
- A completely unspecified structure, such as in Dung’s abstract framework for argumentation (1995).

Conflict, Attack, Counterargument

The notion of conflict (Counterargument or Attack) between arguments is typically discussed discriminating three cases:

- **Rebutting attacks**: arguments with contradictory conclusions.
- **Assumption attack**: attacking non-provability assumptions.
- **Undercutting attacks**: an argument that undermines some intermediate step (inference rule) of another argument.
Rebutting and assumption attacks

Rebutting is symmetric, e.g.:
'Tweety flies because it is a bird'
versus
'Tweety doesn’t fly because it is a penguin'.

Assumption attack:
'Tweety flies because it is a bird and it is not provable that Tweety is a penguin' versus
'Tweety is a penguin'.

Undercutting attack

An argument challenges the connection between the premises and the conclusion.

Tweety flies because all the birds I’ve seen fly
I’ve seen Opus; it is a bird and it doesn’t fly
These types of attack could be *direct* and *indirect*.

**Direct attack**

\[ \neg p \rightarrow p \]

**Indirect attack**

\[ \neg p \rightarrow s \]

The notion of conflict does not embody any form of *comparison*; this is another element of AS.

Defeat has the form of a binary relation between arguments, standing for

- ‘attacking and not weaker’ (defeat)
- ‘attacking and stronger’ (strict defeat)

Terminology varies: ‘defeat’ (Simari, 1989; Prakken & Sartor, 1997), ‘attack’ (Dung, 1995; Bondarenko et al, 1997) and ‘interference’ (Loui, 1998).
Defeat: Comparing Arguments

Argumentation systems vary in their grounds for evaluation of arguments. One common criterion is the specificity principle, which prefers arguments based on the most specific defaults.

\[ \text{bird}(\text{opus}) \rightarrow \text{flies}(\text{opus}) \]
\[ \text{flies}(\text{opus}) \leq \text{bird}(\text{opus}), \text{broken}_\text{wing}(\text{opus}) \]

\{ A, \text{flies}(\text{opus}) \} \leq \{ B, \neg\text{flies}(\text{opus}) \}

Defeat: Comparing Arguments

However, it has been argued that specificity is not a general principle of commonsense reasoning, but rather a standard that might (or might not) be used.

Some researchers even claim that general, domain-independent principles of defeat do not exist, or are very weak.

Some even argue that the evaluation criteria are part of the domain theory, and should also be debatable.

What do you think?
Defeat: comparing arguments

- In Simari&Loui’s framework, specificity is used as a default, but it is ‘modular’: any other preference relation defined among arguments could be used.

- In Dung’s, defeat is an abstract notion, left undefined.

- In Bondarenko’s framework, defeat is limited to attack between arguments (there is no preference at all!)

- Other comparison criteria are possible…

Defeat is basically a binary relation on a set of args.

But … it just tells us something about two arguments, not about a dispute (that may involve many args.)

A common situation is reinstatement as in the example below (where an argument \( C \) reinstates an argument \( A \) by defeating argument \( B \))
Status of Arguments

- The last element in our ontology comes into play... the definition of *Status of Arguments*.
- This notion is the actual output of most Arg.Sys and arguments are divided into (at least) two classes:
  - Arguments with which a dispute can be *won*
  - Arguments with which a dispute can be *lost*
  - Arguments that leave the dispute *undecided*
- Usual terminology: *justified* or *warranted* vs. *defeated* or *overruled* vs. *defensible*, etc.

Status of arguments

- Status of arguments can be computed either in *declarative* or *procedural* form.
- In the declarative form usually requires **fixed-point definitions**, and establishes certain sets of arguments as acceptable (in the context of a set of premises and a evaluation criteria) but **without defining a procedure** for testing whether a given argument is a member of this set.
- *Procedural form* amounts to defining such a procedure for acceptability.
Status of arguments

- **Declarative form of argumentation**
- **Procedural form of argumentation**

- Argumentation-theoretic semantics
- Proof Theory

---

Model-theoretic Semantics

- Default logic was initially criticized by the lack of a model-theoretic semantics...
- Several researchers argued that NMR needs a different kind of semantics than model theory suggesting an argumentation-theoretic semantics.
- Model theory provides meaning to logical languages by defining how the world would be if an expression with these symbols would be true.
- *Should this be the case for argumentative systems...?*
Some researchers (e.g. Pollock, Vreeswijk, Loui) argue that the meaning of defaults should not be found in a correspondence with reality, but in their role in dialectical inquiry.

This approach goes as follows: since the central notions of defeasible reasoning are not propositional, then the semantics should also be different, i.e., an argumentation-theoretic semantics should be defined.

Defeasible rules “premises ⇒ conclusion” induce a burden of proof, rather than a correspondence between a proposition and the world.

Argumentation-theoretic semantics tries to capture sets of arguments that are as large as possible, and defend themselves against attacks on their members.
Argument-based Semantics

- Which conditions on sets of arguments should be satisfied?
- We will assume as background
  - A set $\text{Args}$ of arguments
  - A binary relation of ‘defeat’ defined over it.

**Def. 1:** Arguments are either *justified* or *not justified*

1. An argument is justified if all arguments defeating it (if any) are not justified.
2. An argument is not justified if it is defeated by an argument that is justified.

Example: Consider three arguments $A$, $B$ and $C$

Argument $A$ and $C$ are justified; argument $B$ is not.
Example: Even cycle

\[
A = \text{"Nixon was a pacifist because he was a quaker"}
\]
\[
B = \text{"Nixon wasn’t a pacifist because he was a republican"}
\]

There are two status assignment that satisfy Def 1

Def. 1: Arguments are either justified or not justified
1. An argument is justified if all arguments defeating it (if any) are not justified.
2. An argument is not justified if it is defeated by an argument that is justified.

Argument-based Semantics

In the literature, two approaches to the solution of this problem can be found.

- **First approach**: changing Def. 1 in such a way that there is always precisely one possible way to assign a status to arguments. Undecided conflicts get the status ‘not justified’.
  
  Allowing unique-status assignment (u.s.a).

- **Second approach**: allowing multiple assignments, defining an argument as ‘genuinely’ justified iff it is justified in all possible assignments.
  
  Allowing multiple-status assignment (m.s.a).
Self-defeating Argument

Another problem with Definition 1

• The role of self-defeating arguments.

Self-defeating arguments are inconsistent with Definition 1

but...

They can be considered as plausible constructions.

The Unique-Status-Assignment Approach

This idea could be presented in two different ways:

- Using a fixed-point operator
- Given a recursive definition of justified argument
Fixed-point Definitions

This approach has been used in several frameworks, e.g., Pollock (1987, 1992), Simari & Loui (1992) and Prakken & Sartor (1997). It is based on the notion of reinstatement, captured by Dung’s definition of acceptability:

**Def. 2: (Acceptability)**
An argument $A$ is acceptable wrt a set $S$ of arguments iff each argument defeating $A$ is defeated by an argument in $S$.

A Fixed-point Operator

However, this notion seems to be not sufficient...

**Def. 3: (Dung’s Grounded Semantics)** Let $Args$ be a set of arguments ordered by a binary relation of defeat, and let $S \subseteq Args$. Then the operator $F$ is defined as follows.

$$F(S) = \{ A \in Args \mid A \text{ is acceptable wrt } S \}$$
A Fixed-point Operator

Dung proves that the operator $F$ has a least fixed point

Def. 4: (Justified Argument) An arg. is justified iff it is a member of the least fixed point of $F$.

Def. 5: (Least fixed point of $F$)

- $F^0 = \emptyset$
- $F^{i+1} = \{ A \in \text{Args} | A \text{ is acceptable wrt } F^i \}$

Propositions

1. All arguments in $\bigcup_{i=0}^{\infty} (F^i)$ are justified.
2. If each argument is defeated by at most a finite number of arguments, then an argument is justified iff it is in $\bigcup_{i=0}^{\infty} (F^i)$.

Consider the previous example:

$F^1 = F(\emptyset) = \{ C \}$

$F^2 = F(F(\emptyset)) = \{ A, C \}$

$F^3 = F(F^2(\emptyset)) = F^2$
Def. 6: (G operator) Let \( \text{Args} \) be a set of arguments ordered by a binary relation of defeat. Then
\[
G(S) = \{ A \in \text{Args} \mid A \text{ is not defeated by any arg. in } S \}
\]

Def. 7: (Levels in justification)
- All arguments are in level 0
- An argument is in at level \((n+1)\) iff it is not defeated by any argument at level \(n\)
- An argument is justified iff there is an \(m\) such that for every \(n \geq m\), the argument is in at level \(n\).

Examples

<table>
<thead>
<tr>
<th>Level</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(A, B)</td>
</tr>
<tr>
<td>1</td>
<td>(A, B)</td>
</tr>
<tr>
<td>2</td>
<td>(A, B)</td>
</tr>
<tr>
<td>3</td>
<td>(A, B)</td>
</tr>
<tr>
<td>4</td>
<td>(A, B)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>IN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(A, B, C)</td>
</tr>
<tr>
<td>1</td>
<td>(C)</td>
</tr>
<tr>
<td>2</td>
<td>(A, C)</td>
</tr>
<tr>
<td>3</td>
<td>(A, C)</td>
</tr>
<tr>
<td>4</td>
<td>(\ldots)</td>
</tr>
</tbody>
</table>
### Infinite defeat chain

Consider an infinite chain of args $A_1, \ldots, A_n$ such that $A_1$ is defeated by $A_2$, $A_2$ is defeated by $A_3$, and so on.

\[ A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \]

The least fixed point of this chain is empty, since no argument is undefeated. Consequently, $F(\emptyset) = \emptyset$.

This example has two other fixed points:

\[
F_1 = \{ A_1, A_3, A_5, A_7, \ldots \} \\
F_2 = \{ A_2, A_4, A_6, A_8, \ldots \}
\]

### Defensible and Overruled Arguments

Consider the following situation:

```
A  B  C
```

\[ B \text{ is not defeated by a justified argument!} \]

"$B$" is called "zombie argument" (Makinson & Schlechta, 1991), or "defensible arguments" (Prakken & Sartor).

**Def 8: (Overruled and defensible arguments)**

- $A$ is overruled iff $A$ is not justified, and $A$ is defeated by a justified argument.
- $A$ is defensible iff $A$ is not justified and $A$ is not overruled.
Defensible and Overruled Arguments

In summary:

- \(\text{Justified}\)
- \(\text{Not Justified}\)
- Properly “Not Justified” = Overruled
- Defensible

Self-defeating arguments

Intuitively, \(B\) should be justified ...
But \(F(\emptyset) = \emptyset\), so neither of them is!

Def. 9: (Levels in justification / modified)
- An argument is \(\text{in}\) at level 0 iff it is not self-defeating.
- An argument is \(\text{in}\) at level \((n+1)\) iff it is \(\text{in}\) at level 0 and it is not defeated by any arg. at level \(n\).
- An argument is \(\text{justified}\) iff there is an \(m\) such that for every \(n \geq m\), the argument is \(\text{in}\) at level \(n\).
Self-defeating Arguments

Apart from Pollock’s refined version of “level-$n$ arguments”, there are other possible solutions to self-defeating arguments:

- Distinguishing a special empty argument which defeats any self-defeating argument (Prakken & Sartor, Vreeswijk).

- Demanding that by construction arguments must be non self-defeating, (Simari & Loui).

Problems with Unique-Status Assignment

There are some problems when evaluating unique-status assignment.

Example: Floating Arguments / Floating Conclusions

The unique-status approach is inherently unable to capture floating arguments and conclusions.
A second way to deal with competing arguments of equal strength is to let them induce two alternative status assignments.

Evaluating outcomes from alternative status assignments let us determine when an argument is justified.

**Def. (Status assignment)** Given a set $S$ of args ordered by a binary defeat relation, an status assignment $sa(S)$ is a function which maps every argument in $S$ into $\{\text{in, out}\}$, such that:

1. $A$ is in iff all args defeating it (if any) are out.
2. $A$ is out if it is defeated by an arg that is in.

**Example**

**Def. (Justification)** Given a set $S$ of arguments ordered by a binary defeat relation, an argument is justified iff it is in in all possible status assignments to $S$. 
Classifying Arguments

Def. : Given a set $S$ of arguments ordered by a binary defeat relation, an argument $A$ is
- justified iff it is ‘$in$’ in all $sa(S)$.
- overruled iff it is ‘$out$’ in all $sa(S)$
- defensible iff it is ‘$out$’ in some $sa(S)$, ‘$in$’ in others.

⇒ Are the two approaches are equivalent?
⇒ The answer is no.

Equivalent?

The unique-status approach says ‘all arguments are defensible’
The multiple-status approach says ‘$C$ is overruled’, and ‘$D$ is justified’
**Status of Conclusions**

Def.: (Status of Conclusions)
- $\phi$ is a justified conclusion iff every status assignment assigns ‘in’ to an arg. with conclusion $\phi$.
- $\phi$ is a defensible conclusion iff $\phi$ is not justified, and a conclusion of a defensible argument.
- $\phi$ is an overruled conclusion iff $\phi$ is not justified or defensible, and a conclusion of an overruled argument.

* Changing the first clause into ‘$\phi$ is a justified conclusion iff $\phi$ is the conclusion of a justified argument’ would make a stronger notion ...

---

**Problems with Multiple-Status Assignment**

- What are the status assignments?
- There are no status assignments!
Comparing the two approaches

- Some researchers say that the difference between the two approaches can be compared with the ‘skeptical’ vs. ‘credulous’ attitude towards drawing defeasible conclusions ...

- m.s.a is more convenient for identifying sets of arguments that are compatible with each other.

- u.s.a considers arguments on an individual basis.

Example

Note that $A$ and $D$ are somehow incompatible; in the unique-assignment approach this notion is (or seems) harder to capture.

- This example has 2 status assignments: \{ $A$, $C$ \} and \{ $B$, $D$ \}
### Outline

- (Very brief) Introduction to Multiagent Systems
- What is argumentation? Fundamentals
- A Case Study: DeLP and its extensions as an argument-based approach to logic programming.
- Argument-based negotiation
- Conclusions

### Deafeasible Logic Programming: DeLP

A *Defeasible Logic Program* (dlp) is a set of facts, strict and defeasible rules denoted $\mathcal{P} = (\Pi, \Delta)$

\[
\begin{align*}
\Pi & \quad \text{Strict Rules} \\
& \quad \{ \text{bird}(X) \leftarrow \text{chicken}(X), \text{chicken}(\text{tina}) \} \\
& \quad \{ \text{bird}(X) \leftarrow \text{penguin}(X), \text{penguin}(\text{opus}) \} \\
& \quad \{ \neg \text{flies}(X) \leftarrow \text{penguin}(X), \text{scared}(\text{tina}) \} \\
\end{align*}
\]

\[
\begin{align*}
\Delta & \quad \text{Defeasible Rules} \\
& \quad \{ \text{flies}(X) \leftarrow \text{bird}(X) \} \\
& \quad \{ \neg \text{flies}(X) \leftarrow \text{chicken}(X) \} \\
& \quad \{ \text{flies}(X) \leftarrow \text{chicken}(X), \text{scared}(X) \} \\
\end{align*}
\]
Def: Let $L$ be a literal and $P = \langle \Pi, \Delta \rangle$ be a program. 
$\langle \mathcal{A}, L \rangle$ is an argument, for $L$, if $\mathcal{A}$ is a set of rules in $\Delta$ such that:

1) There exists a defeasible derivation of $L$ from $\Pi \cup \mathcal{A}$;
2) The set $\Pi \cup \mathcal{A}$ is non-contradictory; and
3) There is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}'$ satisfies 1) and 2).

\[
\begin{align*}
\text{buy\_shares}(X) & \leftarrow \text{good\_price}(X) \\
\neg \text{buy\_shares}(X) & \leftarrow \text{good\_price}(X), \text{risky}(X) \\
\text{risky}(X) & \leftarrow \text{in\_fusion}(X, Y) \\
\neg \text{risky}(X) & \leftarrow \text{in\_fusion}(X, Y), \text{strong}(Y) \\
\text{good\_price}(\text{acme}) & \\
\text{in\_fusion}(\text{acme, estron}) & \\
\text{strong}(\text{estron}) & \\
\{ & \\
\neg \text{buy\_shares}(\text{acme}) & \leftarrow \text{good\_price}(\text{acme}), \text{risky}(\text{acme}) \\
\text{risky}(\text{acme}) & \leftarrow \text{in\_fusion}(\text{acme, enron}) \\
\} & \\
, & \\
\neg \text{buy\_shares}(\text{acme}) & \\
\} & \end{align*}
\]
\( \langle S, Q \rangle \) is a subargument of \( \langle A, L \rangle \) if \( S \) is an argument for \( Q \) and \( S \subseteq A \)

\[
A = \{ \neg \text{buy} \_\text{shares} (\text{acme}) \rightarrow \text{good} \_\text{price} (\text{acme}), \text{risky} (\text{acme}), \\
\text{risky} (\text{acme}) \rightarrow \text{in} \_\text{fusion} (\text{acme}, \text{enron}) \}
\]

\[
S = \{ \text{risky} (\text{acme}) \rightarrow \text{in} \_\text{fusion} (\text{acme}, \text{enron}) \}
\]

\[ P \cup \{ \text{risky} (\text{acme}), \neg \text{risky} (\text{acme}) \} \]

is a contradictory set
Argument Comparison: Generalized Specificity

**Def:** Let $\mathcal{P} = (\Pi, \Delta)$ be a program, let $\Pi_G$ be the set of strict rules in $\Pi$ and let $\mathcal{F}$ be the set of all literals that can be defeasibly derived from $\mathcal{P}$. Let $\langle A_1, L_1 \rangle$ and $\langle A_2, L_2 \rangle$ be two arguments built from $\mathcal{P}$, where $L_1, L_2 \in \mathcal{F}$. Then $\langle A_1, L_1 \rangle$ is **strictly more specific** than $\langle A_2, L_2 \rangle$ if:

1. For all $H \subseteq \mathcal{F}$, if there exists a defeasible derivation $\Pi_G \cup H \cup A_1 \vdash L_1$ while $\Pi_G \cup H \not\vdash L_1$ then $\Pi_G \cup H \cup A_1 \vdash L_2$, and

2. There exists $H' \subseteq \mathcal{F}$ such that there exists a defeasible derivation $\Pi_G \cup H' \cup A_2 \vdash L_2$ and $\Pi_G \cup H' \not\vdash L_2$ but $\Pi_G \cup H' \cup A_1 \not\vdash L_1$


---

Defeaters

An argument $\langle B, P \rangle$ is a **defeater** for $\langle A, L \rangle$ if $\langle B, P \rangle$ is a counter-argument $\langle A, L \rangle$ that attacks a subargument $\langle S, Q \rangle$ de $\langle A, L \rangle$ and one of the following conditions holds:

(a) $\langle B, P \rangle$ is better than $\langle S, Q \rangle$ (proper defeater), or

(b) $\langle B, P \rangle$ is not comparable to $\langle S, Q \rangle$ (blocking defeater)
Given $\mathcal{P} = (\Pi, \Delta)$, and $\langle A_0, L_0 \rangle$ an argument obtained from $\mathcal{P}$. An argumentation line for $\langle A_0, L_0 \rangle$ is a sequence of arguments obtained from $\mathcal{P}$, denoted $\Lambda = [\langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots]$ where each element in the sequence $\langle A_i, h_i \rangle$, $i > 0$ is a defeater for $\langle A_{i-1}, h_{i-1} \rangle$.

Given an argumentation line $\Lambda = [\langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots]$, the subsequence $\Lambda_S = [\langle A_0, L_0 \rangle, \langle A_1, L_1 \rangle, \ldots]$ contains supporting arguments and $\Lambda_I = [\langle A_1, L_1 \rangle, \langle A_3, L_3 \rangle, \ldots]$ are interfering arguments.
Given an argumentation line $\Lambda = [(A_0, L_0), (A_1, L_1), \ldots]$, the subsequence $\Lambda_S = [(A_0, L_0), (A_2, L_2), \ldots]$ contains supporting arguments and $\Lambda_I = [(A_1, L_1), (A_3, L_3), \ldots]$ are interfering arguments.

### Acceptable Argumentation Line

Given a program $P = (\Pi, \Delta)$, an argumentation line $\Lambda = [(A_0, L_0), (A_1, L_1), \ldots]$ will be acceptable if:

1. $\Lambda$ is a finite sequence.
2. The sets $\Lambda_S$ of supporting arguments is concordant, and the set $\Lambda_I$ of interfering arguments is concordant.
3. There is no argument $(A_k, L_k)$ in $\Lambda$ that is a subargument of a preceding argument $(A_i, L_i)$, $i < k$.
4. For all $i$, such that $(A_i, L_i)$ is a blocking defeater for $(A_{i+1}, L_{i+1})$, if there exists $(A_{i+1}, L_{i+1})$ then $(A_{i+1}, L_{i+1})$ is a proper defeater for $(A, L)$ (i.e., $(A, L)$ could not be blocked).
A literal $L$ will be warranted if there is an argument $\langle A, L \rangle$ built from $P$, and that argument has a dialectical tree whose root node is marked $U$. That is, argument $\langle A, L \rangle$ is an argument for which all the possible defeaters have been defeated. We will say that $A$ is a warrant for $L$. \[ \mathcal{T} \langle A, L \rangle \]

Marking of a Dialectical Tree

Given a program $P = (\Pi, \Delta)$, a literal $L$ will be warranted if there is an argument $\langle A, L \rangle$ built from $P$, and that argument has a dialectical tree whose root node is marked $U$. That is, argument $\langle A, L \rangle$ is an argument for which all the possible defeaters have been defeated. We will say that $A$ is a warrant for $L$. \[ \mathcal{T}^* \langle A, L \rangle \]
Answers in DeLP

Given a program $\mathcal{P} = (\Pi, \Delta)$, and a query for $L$ the possible answers are:

- **YES**, if $L$ is warranted.
- **NO**, if $\neg L$ is warranted.
- **UNDECIDED**, if neither $L$ nor $\neg L$ are warranted.
- **UNKNOWN**, if $L$ is not in the language of the program.

DeLP : extensions

- Recently extensions of DeLP have been developed:
  - **P-DeLP** (Chesñevar et. al, 2004): aims at modelling reasoning under uncertainty (e.g. possibilistic reasoning).
  - **O-DeLP** (Capobianco et. al, 2004): aims at modelling reasoning for agents in changing environments.
Argument-based Recommenders

NL assessment using arguments
**ODeLP-based agent architecture**

- Observations
- Defeasible rules
- Dialectical base
- Updating mechanism

**ODeLP inference engine**

**P-DeLP in an agent’s reasoning module**

Sample rules:

- When there is pump clog, fuel is not ok:
  
  $$(\neg fuel\_ok \leftarrow pump\_clog, 1)$$

- When there is heat, usually engine is not ok.
  
  $$(\neg engine\_ok \leftarrow heat, 0.95)$$

**Engine has 3 switches on**

- Oil Pump
- Fuel Pump
- Motor

**There is heat**

- sw1
- sw2
- sw3

**Is the engine ok?**

**Speed: 03**
Agent Reasoning Module

Facts
Rules
P-DeLP program
Dialectical Base
Updating mechanism
P-DeLP Inference engine

Sensor input (perception)
queries
answers

Environment (e.g. engine)
Other Agent (e.g. supervisor agent)

Query: engine_ok?
Answer: No (0.3)

\( \langle A_1, \text{engine\_ok}, 0.3 \rangle \)
\( \langle A_2, \neg\text{fuel\_ok}, 0.6 \rangle \)
\( \langle A_3, \neg\text{low\_speed}, 0.6 \rangle \)
\( \langle A_4, \neg\text{fuel\_ok}, 0.6 \rangle \)
\( \langle A_5, \neg\text{engine\_ok}, 0.95 \rangle \)

Second Part