Discontinuity-Free Decision Support with Quantitative Argumentation Debates

A. Rago [1], F. Toni [1], M. Aurisicchio [1] & P. Baroni [2]

1. Imperial College London
2. Universita` degli Studi di Brescia

Cardiff Argumentation Forum

6th July 2016

Presentation Overview

1. Background
    - IBIS Charts and QuAD Frameworks
    - QuAD Algorithm/Semantics

2. Research Summary
    - Motivation for DF-QuAD
    - DF-QuAD Algorithm
    - Comparison of the DF-QuAD and QuAD Algorithms
    - Properties Not Held by QuAD
    - Properties Shared with QuAD
    - Relationship to Abstract Argumentation
    - Reverse Engineering Functionality
    - Applications of QuAD Frameworks

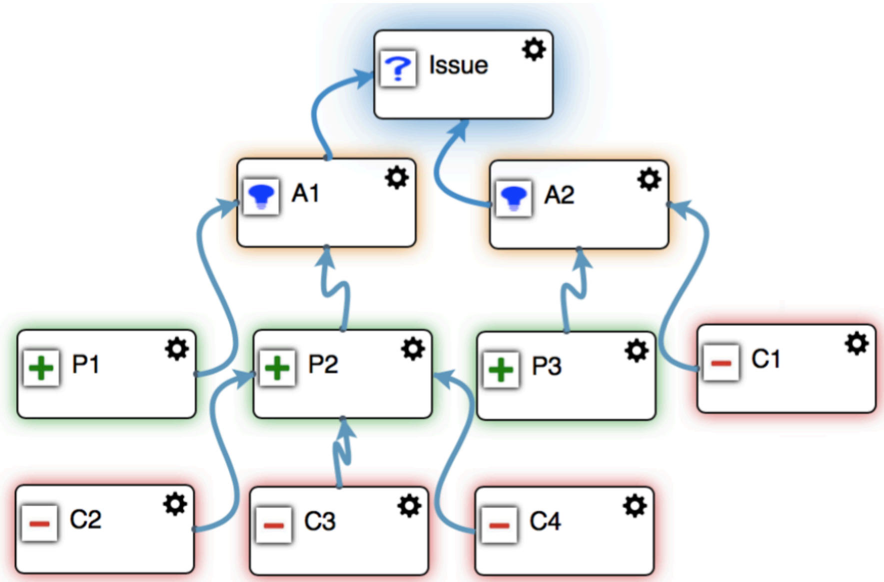
3. Future Work

4. Conclusions

Background – IBIS Charts & QuAD Frameworks

- IBIS (Issue Based Information System) charts [Kunz and Rittel, 1970].
- QuAD (Quantitative Argumentation Debate) frameworks [Baroni et al. 2015].
  - Special types of IBIS trees with base scores for nodes.

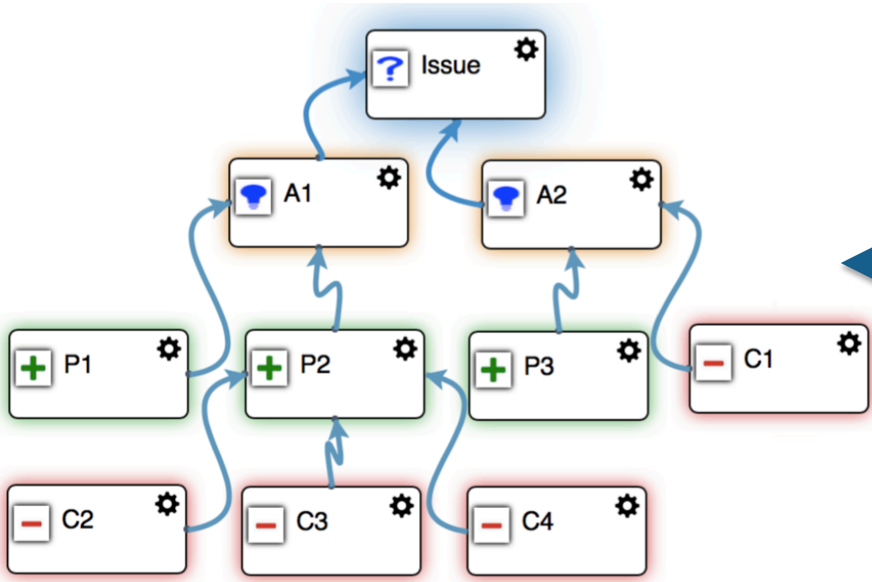
QuAD Framework [www.arganddec.com]



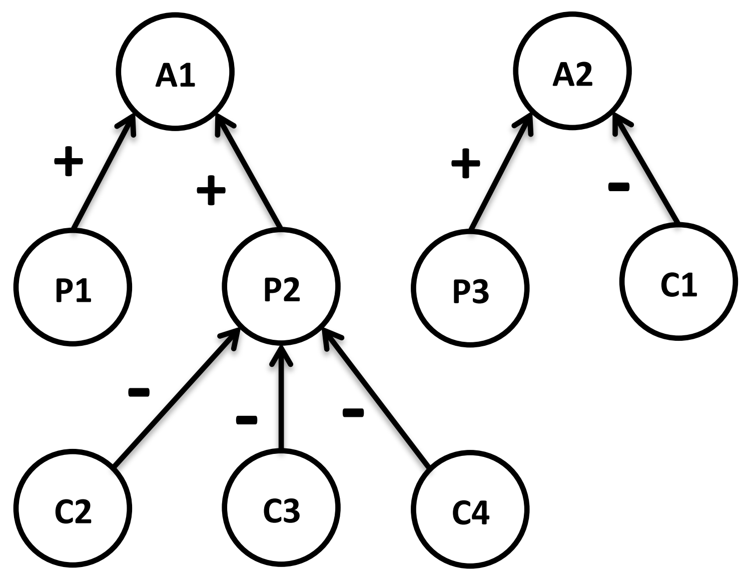
Background – IBIS Charts & QuAD Frameworks

- IBIS (Issue Based Information System) charts [Kunz and Rittel, 1970].
- QuAD (Quantitative Argumentation Debate) frameworks [Baroni et al. 2015].
  - Special types of IBIS trees with base scores for nodes.
- Correspond to BAFs (Bipolar Argumentation Frameworks) [Cayrol and Lagasquie-Schiex, 2005].

QuAD Framework [www.arganddec.com]



BAF Framework



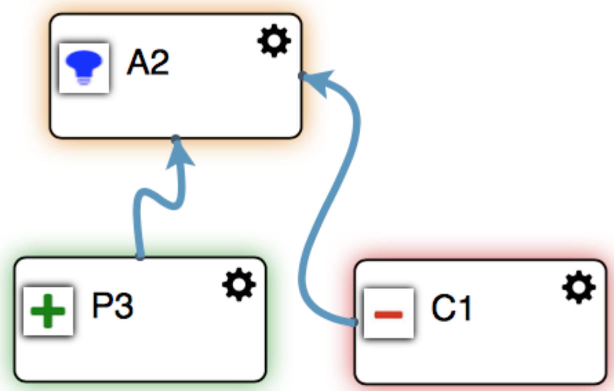
Background – QuAD Algorithm/Semantics

- Base scores are used by the QuAD algorithm to calculate each node's overall *strength*.
- Base scores and strengths are in [0,1].
- Strength is a form of *gradual acceptance* [Cayrol and Lagasquie-Schiex, 2005].

Background – QuAD Algorithm/Semantics

- Base scores are used by the QuAD algorithm to calculate each node's overall *strength*.
- Base scores and strengths are in [0,1].
- Strength is a form of *gradual acceptance* [Cayrol and Lagasquie-Schiex, 2005].
- Firstly, attacking and supporting components (v_a and v_s) are calculated for each node.

$$v_s = \mathcal{F}_{supp}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^+(A2))) \qquad v_a = \mathcal{F}_{att}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^-(A2)))$$



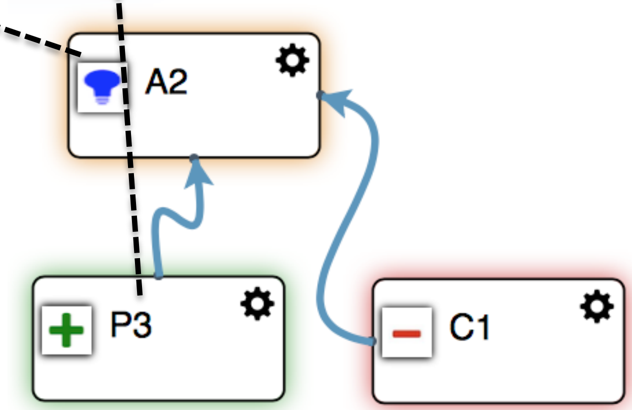
Imperial College London

Background – QuAD Algorithm/Semantics

- Base scores are used by the QuAD algorithm to calculate each node's overall *strength*.
- Base scores and strengths are in [0,1].
- Strength is a form of *gradual acceptance* [Cayrol and Lagasquie-Schiex, 2005].
- Firstly, attacking and supporting components (v_a and v_s) are calculated for each node.

$$v_s = \mathcal{F}_{supp}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^+(A2)))$$

$$v_a = \mathcal{F}_{att}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^-(A2)))$$

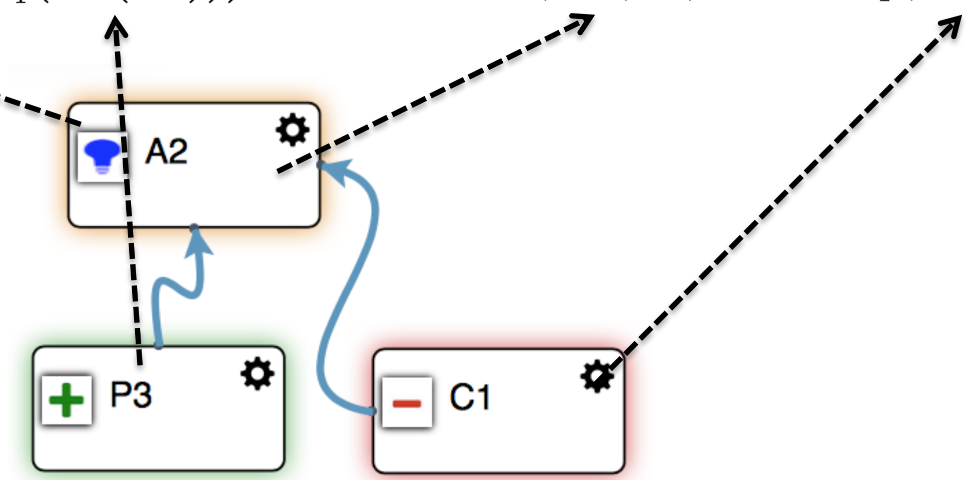


Imperial College London

Background – QuAD Algorithm/Semantics

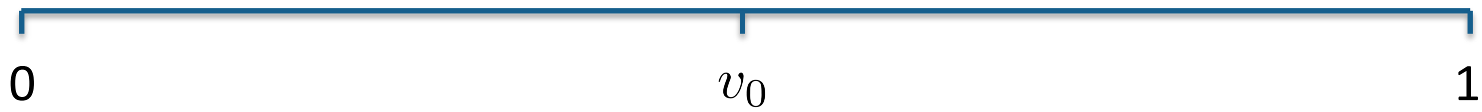
- Base scores are used by the QuAD algorithm to calculate each node's overall *strength*.
- Base scores and strengths are in [0,1].
- Strength is a form of *gradual acceptance* [Cayrol and Lagasquie-Schiex, 2005].
- Firstly, attacking and supporting components (v_a and v_s) are calculated for each node.

$$v_s = \mathcal{F}_{supp}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^+(A2))) \qquad v_a = \mathcal{F}_{att}(\mathcal{BS}(A2), SEQ_{\mathcal{SF}_1}(\mathcal{R}^-(A2)))$$



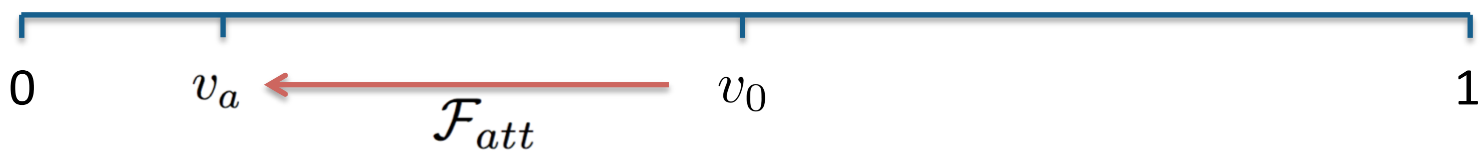
Background – QuAD Algorithm/Semantics

- Recursive formulae are used for attacking and supporting components.



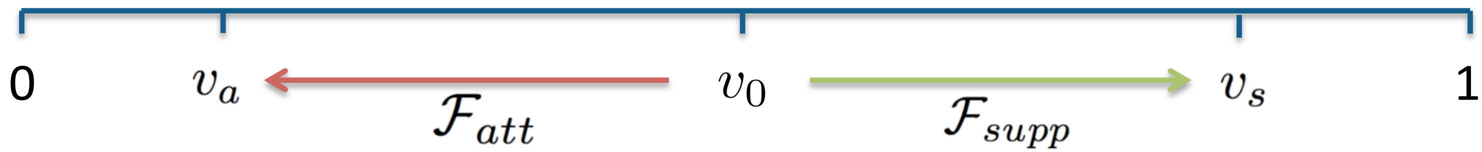
Background – QuAD Algorithm/Semantics

- Recursive formulae are used for attacking and supporting components.



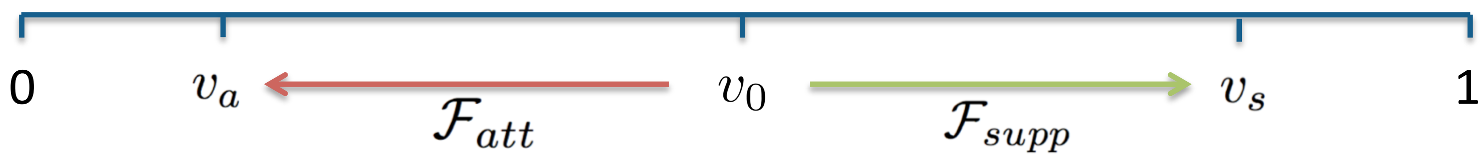
Background – QuAD Algorithm/Semantics

- Recursive formulae are used for attacking and supporting components.



Background – QuAD Algorithm/Semantics

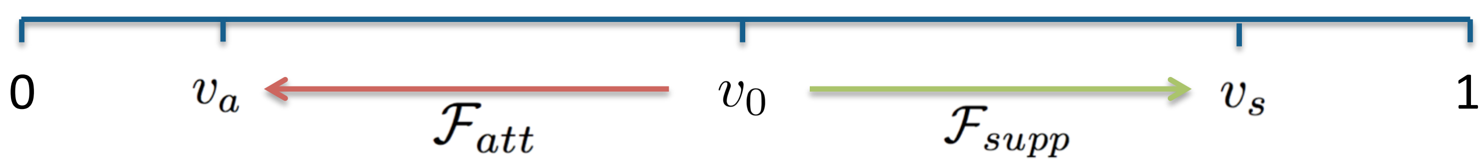
- Recursive formulae are used for attacking and supporting components.



- If the set of attacker/supporter strengths is {} or a set of zeros it is considered **ineffective**.

Background – QuAD Algorithm/Semantics

- Recursive formulae are used for attacking and supporting components.



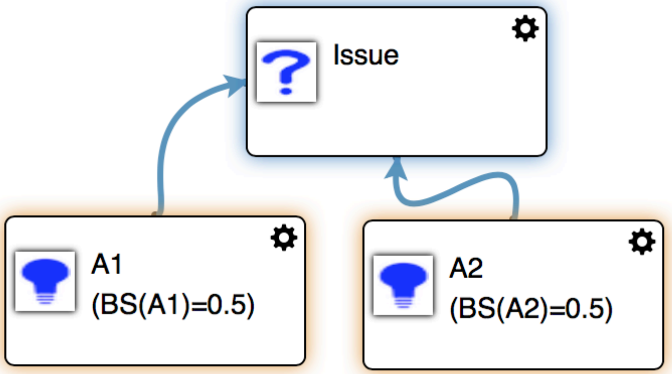
- If the set of attacker/supporter strengths is {} or a set of zeros it is considered **ineffective**.

- The aggregating function then determines the strength in the [0,1] range:

v_a	v_s	Strength
Effective	Ineffective	v_a
Ineffective	Effective	v_s
Ineffective	Ineffective	v_0
Effective	Effective	$(v_a + v_s) / 2$

Research Summary – Motivation for DF-QuAD

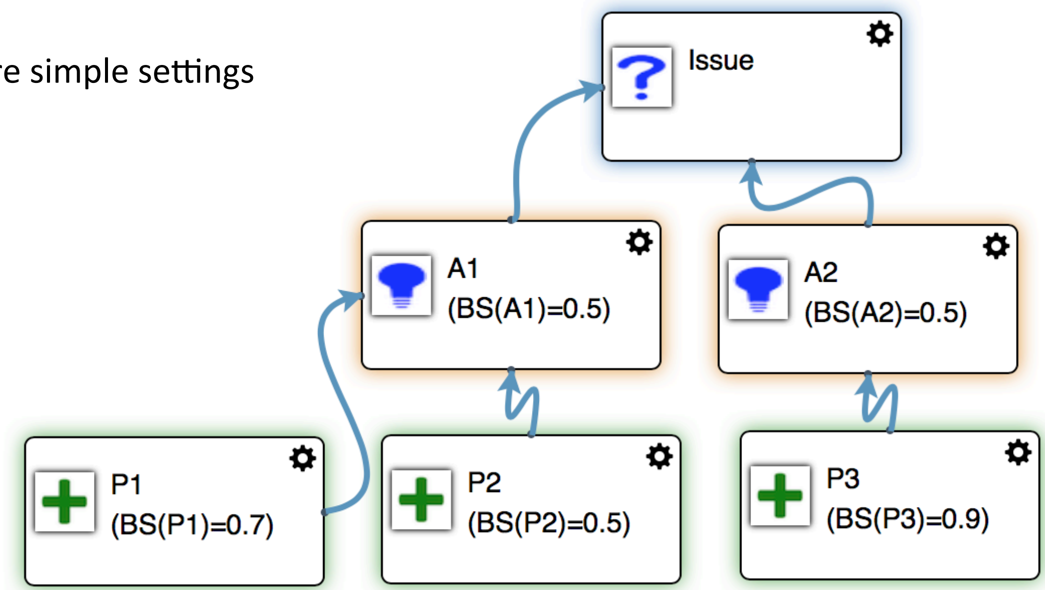
- Engineering Design setting:
  - Issue – Which is the best method for controlling the ventilation of a dining room?
  - Answer 1 – Building management control
  - Answer 2 – User control



Research Summary – Motivation for DF-QuAD

- Engineering Design setting:
  - Issue – Which is the best method for controlling the ventilation of a dining room?
  - Answer 1 – Building management control
  - Answer 2 – User control

- Pro arguments are added at **Stage 1**:
  - Pro 1 – Energy is saved
  - Pro 2 – Elderly occupants require more simple settings
  - Pro 3 – User satisfaction is increased



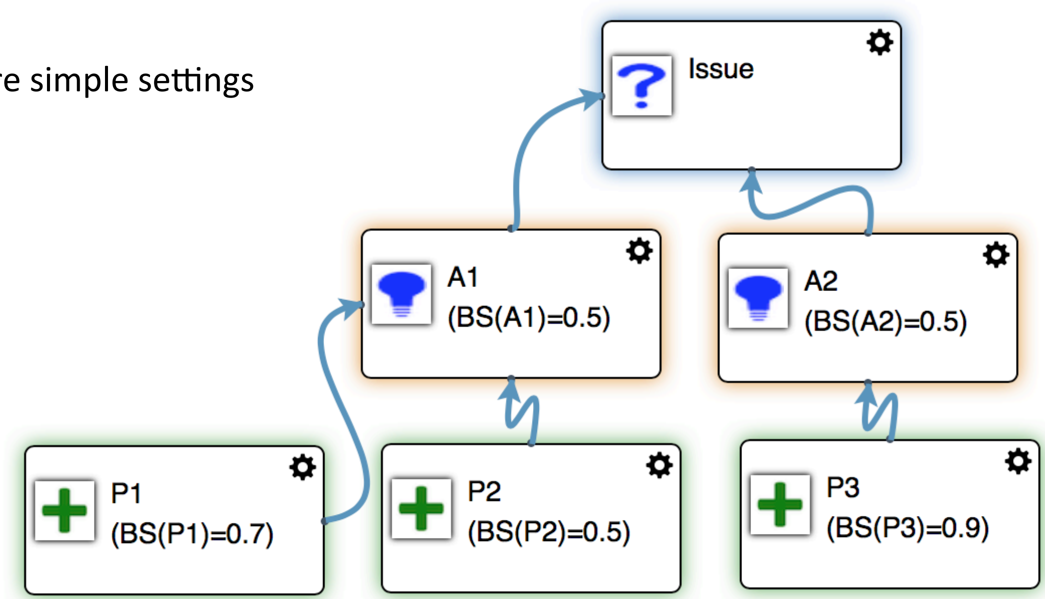
Research Summary – Motivation for DF-QuAD

- Engineering Design setting:
  - Issue – Which is the best method for controlling the ventilation of a dining room?
  - Answer 1 – Building management control
  - Answer 2 – User control

- Pro arguments are added at **Stage 1**:
  - Pro 1 – Energy is saved
  - Pro 2 – Elderly occupants require more simple settings
  - Pro 3 – User satisfaction is increased

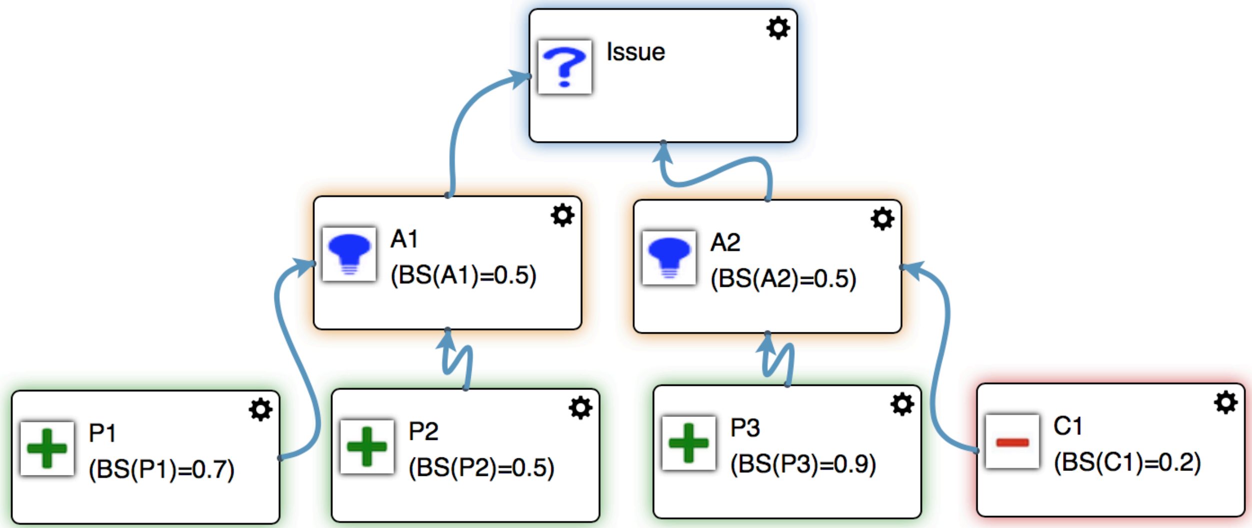
- QuAD Algorithm:

Answer	Strength at Stage 1
A1	0.925
A2	0.950



Research Summary – Motivation for DF-QuAD

- At **Stage 2**, a con argument attacking A2 is then added:
  - Con 1 – User negligence can lead to losses

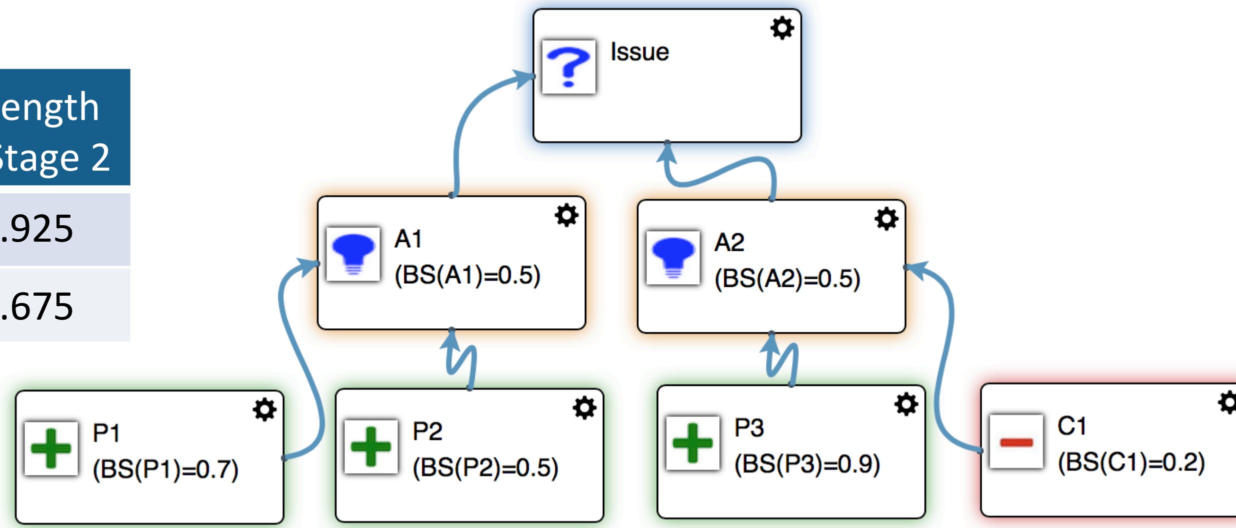


Imperial College London

Research Summary – Motivation for DF-QuAD

- At **Stage 2**, a con argument attacking A2 is then added:
  - Con 1 – User negligence can lead to losses
- QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.675



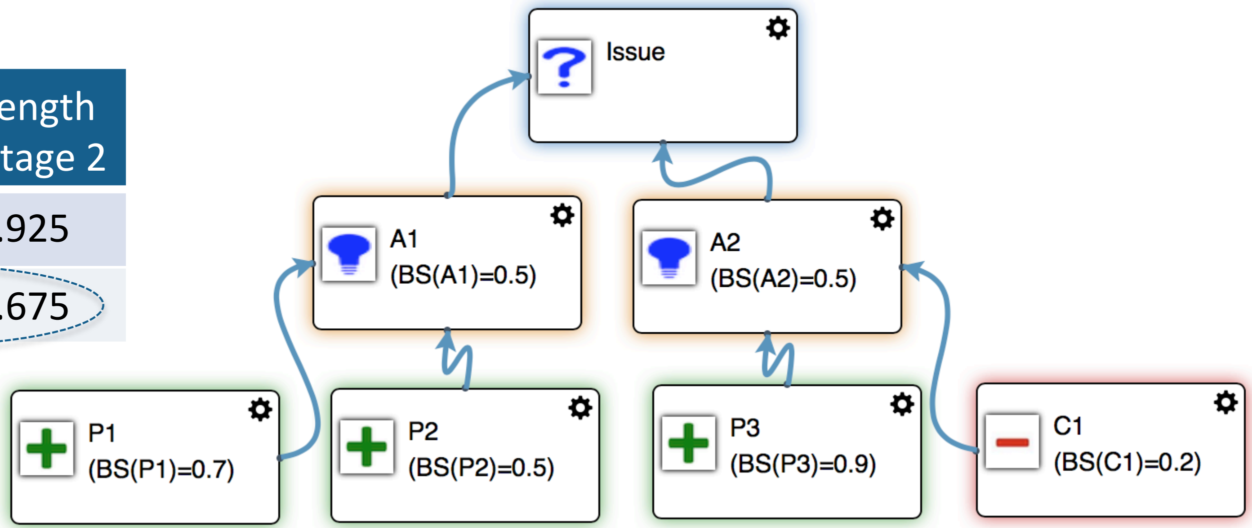
Imperial College London

Research Summary – Motivation for DF-QuAD

- At **Stage 2**, a con argument attacking A2 is then added:
  - Con 1 – User negligence can lead to losses

- QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.675

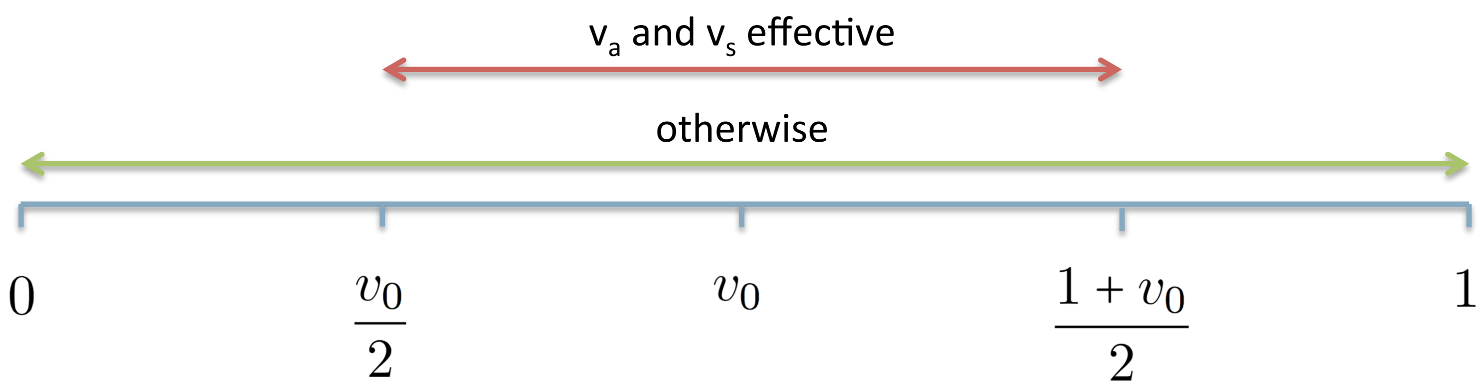


- Large drop in A2's strength between Stage 1 and 2 is disproportionate in some settings, e.g. Engineering Design.

- In other settings, e.g. E-Democracy, this may not be the case.

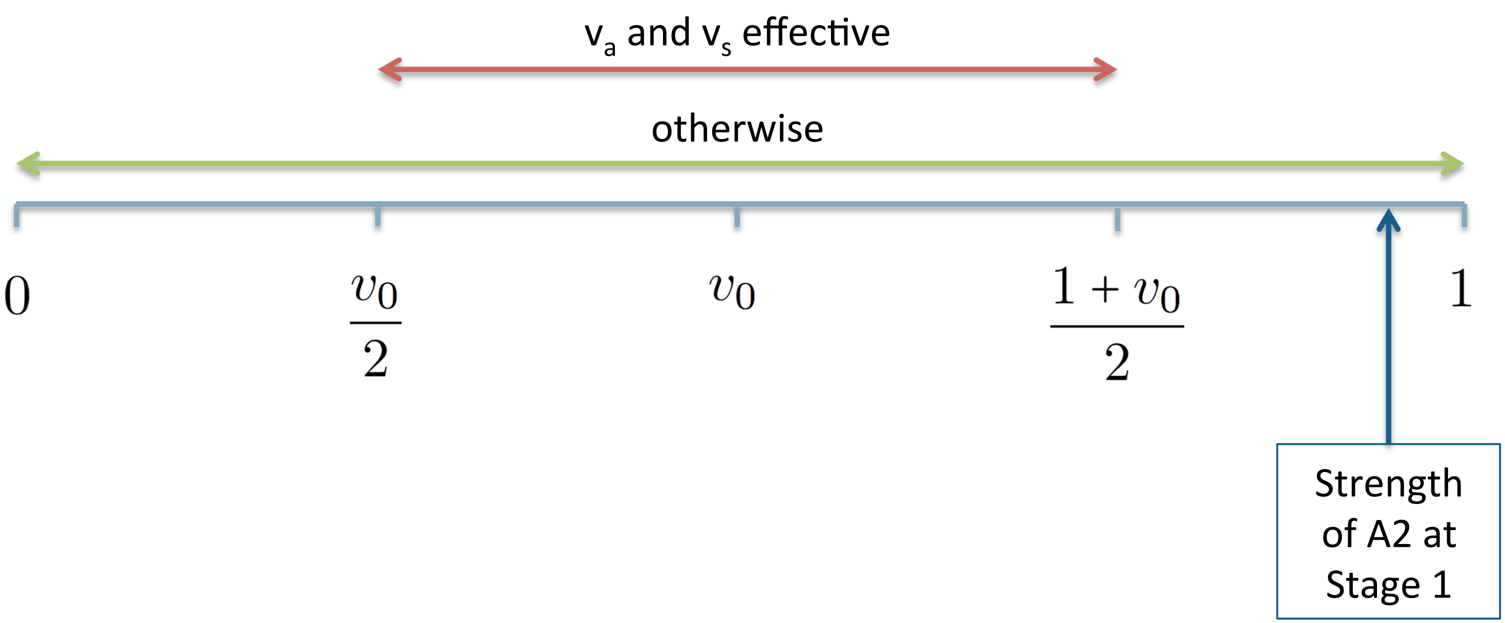
Research Summary – Motivation for DF-QuAD

- The aggregating function's range in one case is a subset of [0,1]:



Research Summary – Motivation for DF-QuAD

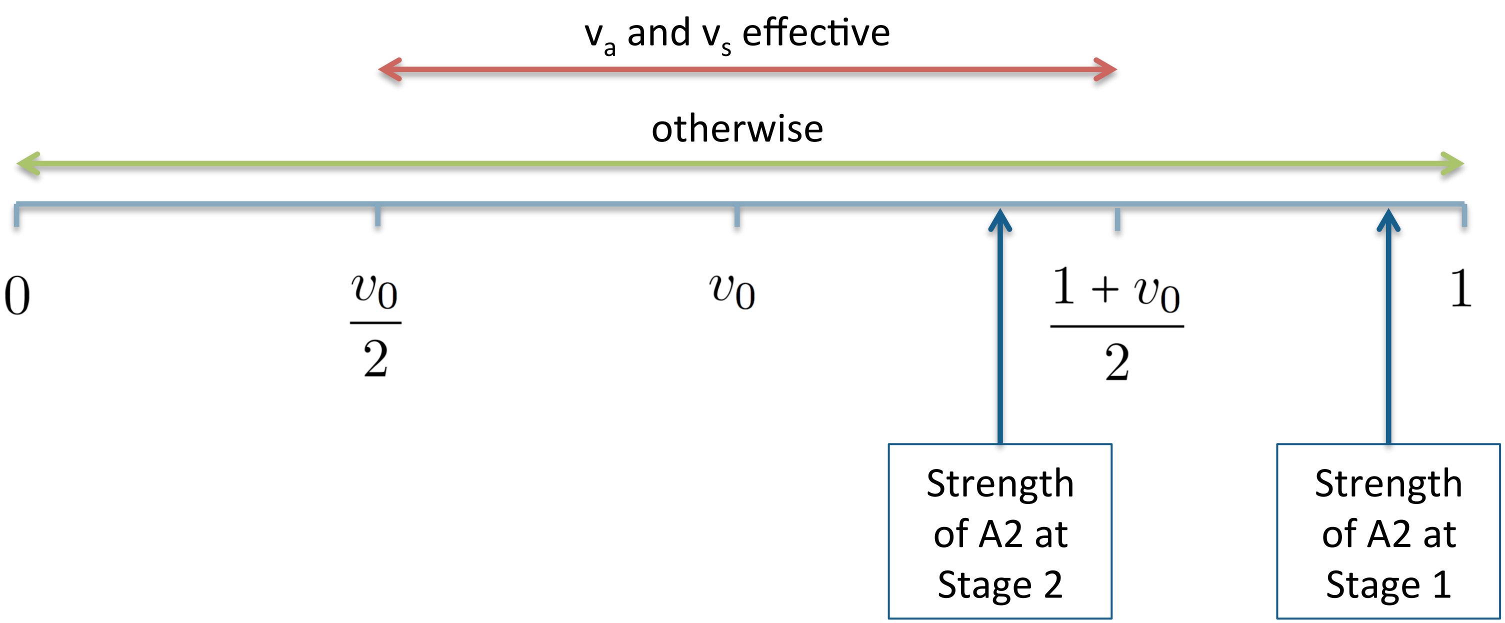
- The aggregating function’s range in one case is a subset of [0,1]:



Imperial College London

Research Summary – Motivation for DF-QuAD

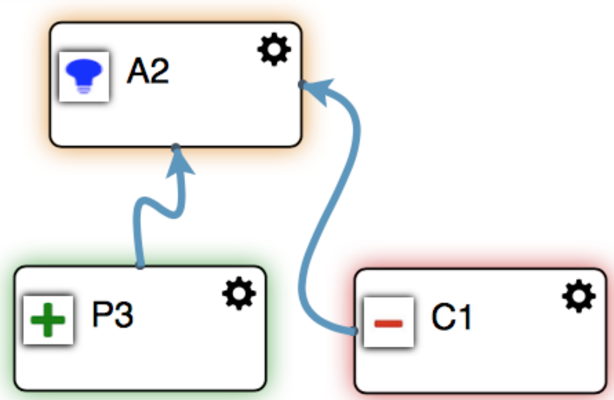
- The aggregating function's range in one case is a subset of [0,1]:



- Discontinuity, leading to counter-intuitive behaviour in some applications.

Research Summary – DF-QuAD Algorithm

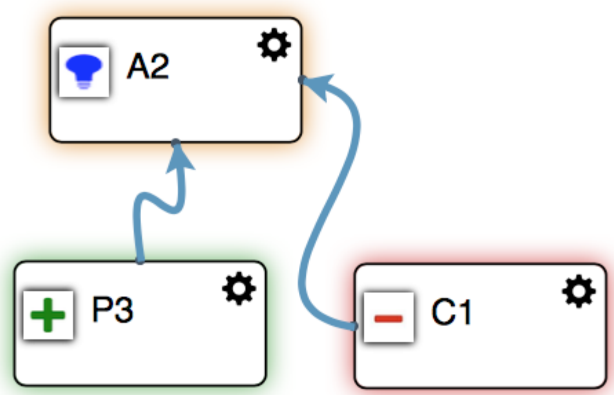
- A new "discontinuity-free" algorithm for QuAD frameworks (DF-QuAD).
- Incorporates many of the same concepts as the QuAD algorithm.
  - Base score and strength in [0,1].



Research Summary – DF-QuAD Algorithm

- A new "discontinuity-free" algorithm for QuAD frameworks (DF-QuAD).
- Incorporates many of the same concepts as the QuAD algorithm.
  - Base score and strength in [0,1].
- A single function used for both the attacking and supporting components.

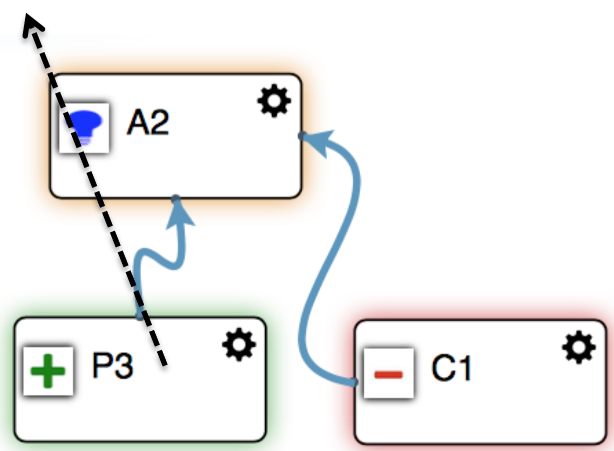
$$v_s = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^+(A2))) \qquad v_a = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^-(A2)))$$



Research Summary – DF-QuAD Algorithm

- A new "discontinuity-free" algorithm for QuAD frameworks (DF-QuAD).
- Incorporates many of the same concepts as the QuAD algorithm.
  - Base score and strength in [0,1].
- A single function used for both the attacking and supporting components.

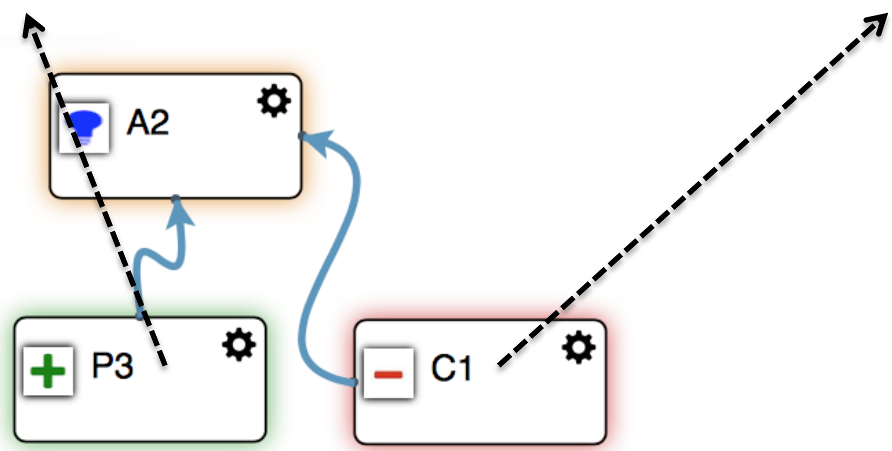
$$v_s = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^+(A2))) \qquad v_a = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^-(A2)))$$



Research Summary – DF-QuAD Algorithm

- A new "discontinuity-free" algorithm for QuAD frameworks (DF-QuAD).
- Incorporates many of the same concepts as the QuAD algorithm.
  - Base score and strength in [0,1].
- A single function used for both the attacking and supporting components.

$$v_s = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^+(A2))) \qquad v_a = \mathcal{F}(SEQ_{\mathcal{SF}_2}(\mathcal{R}^-(A2)))$$



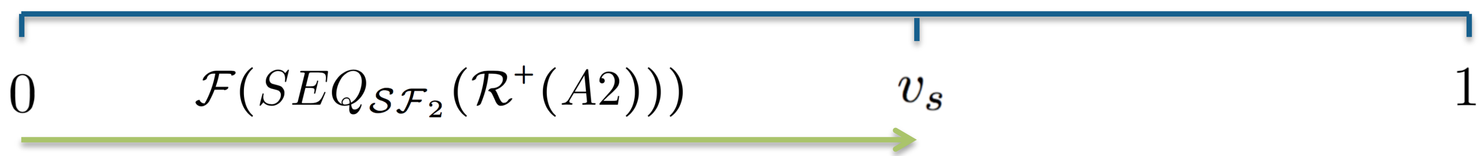
Research Summary – DF-QuAD Algorithm

- The function recursively calculates values for attacking and supporting components:

0 1

Research Summary – DF-QuAD Algorithm

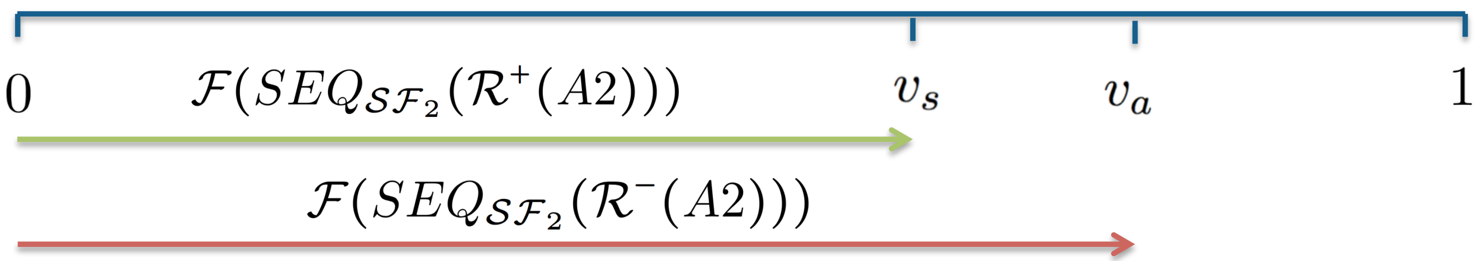
- The function recursively calculates values for attacking and supporting components:



Imperial College London

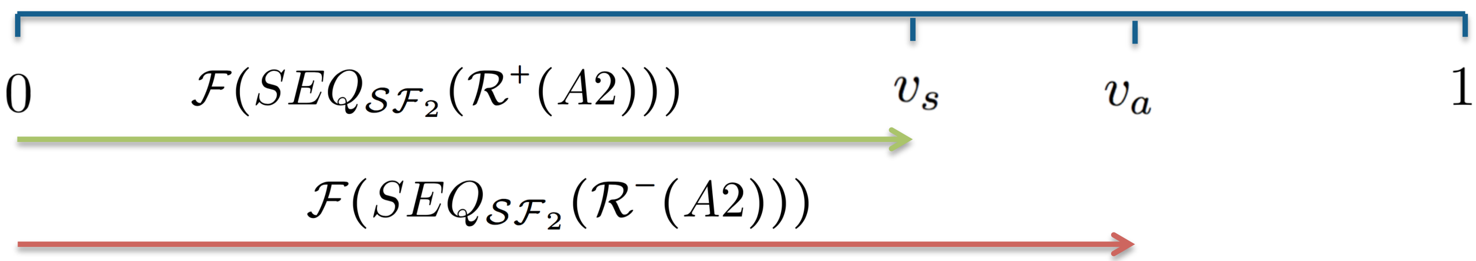
Research Summary – DF-QuAD Algorithm

- The function recursively calculates values for attacking and supporting components:



Research Summary – DF-QuAD Algorithm

- The function recursively calculates values for attacking and supporting components:

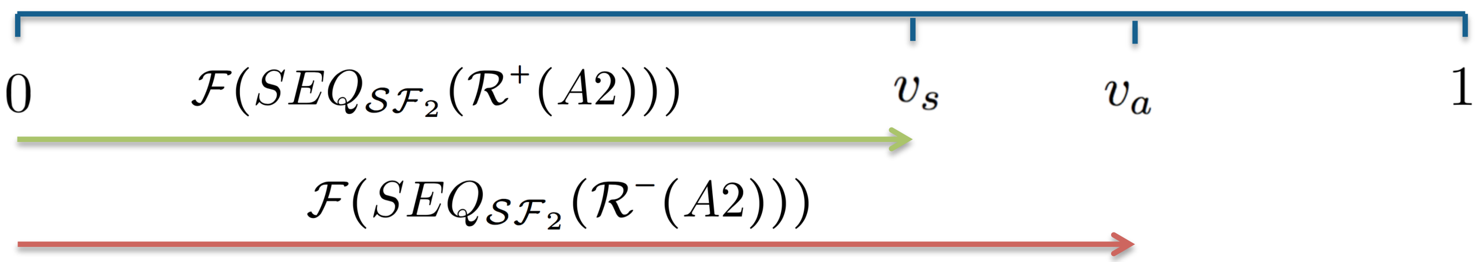


- The combination function then uses the base score and the **difference** between the attacking/supporting components to calculate the strength.

$$
\begin{aligned}
c(v_0, v_a, v_s) &= v_0 - v_0 \cdot |v_s - v_a| && \text{if } v_a \geq v_s \\
c(v_0, v_a, v_s) &= v_0 + (1 - v_0) \cdot |v_s - v_a| && \text{if } v_a < v_s
\end{aligned}
$$

Research Summary – DF-QuAD Algorithm

- The function recursively calculates values for attacking and supporting components:



- The combination function then uses the base score and the **difference** between the attacking/supporting components to calculate the strength.

$$
\begin{aligned}
c(v_0, v_a, v_s) &= v_0 - v_0 \cdot |v_s - v_a| && \text{if } v_a \geq v_s \\
c(v_0, v_a, v_s) &= v_0 + (1 - v_0) \cdot |v_s - v_a| && \text{if } v_a < v_s
\end{aligned}
$$

- The strength is not restricted to a subset of [0,1] when attackers and supporters are effective.

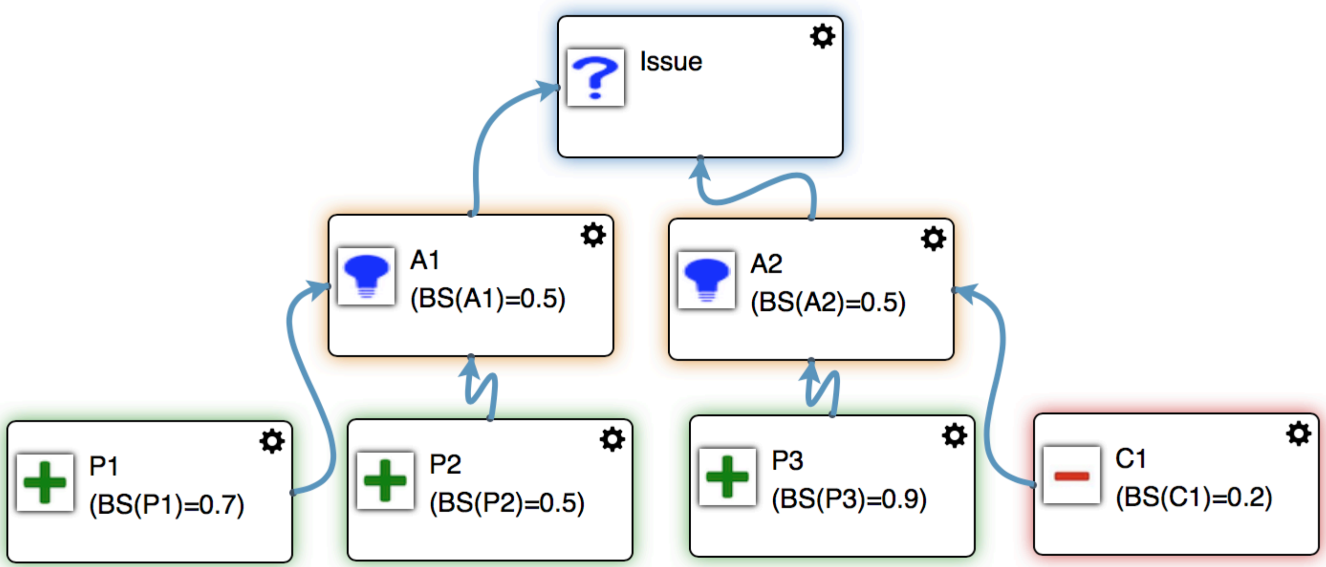
- Identical results to the QuAD algorithm when attackers or supporters are ineffective.

Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

- Engineering Design setting as used for QuAD.

QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.675



Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

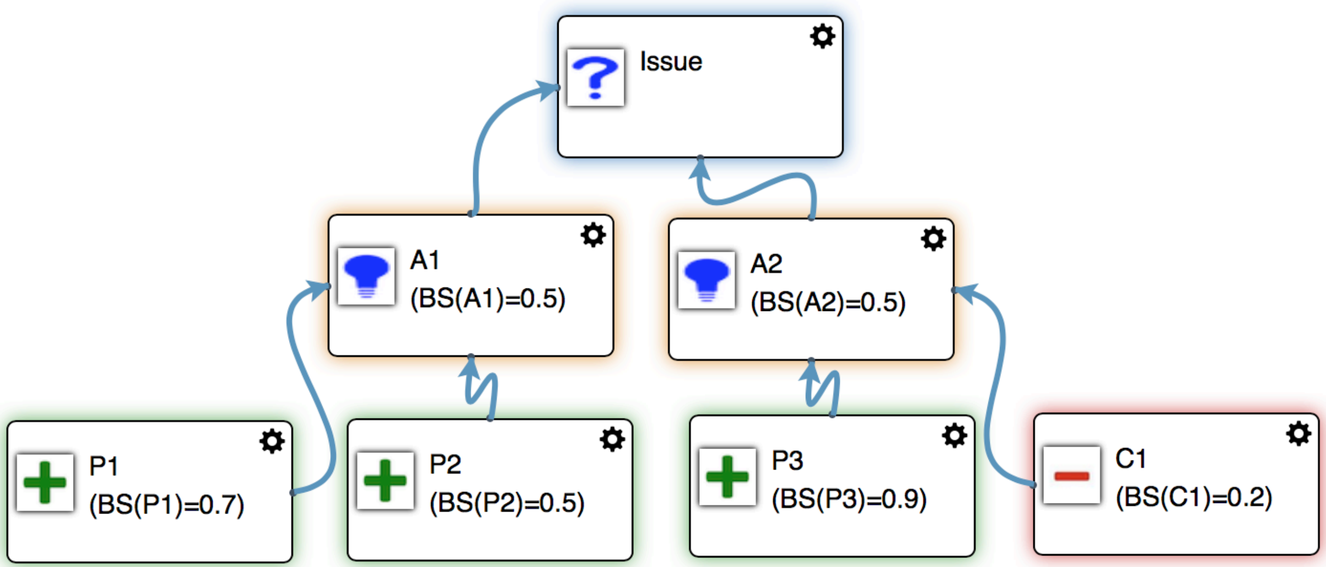
- Engineering Design setting as used for QuAD.

QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.675

DF-QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.850



Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

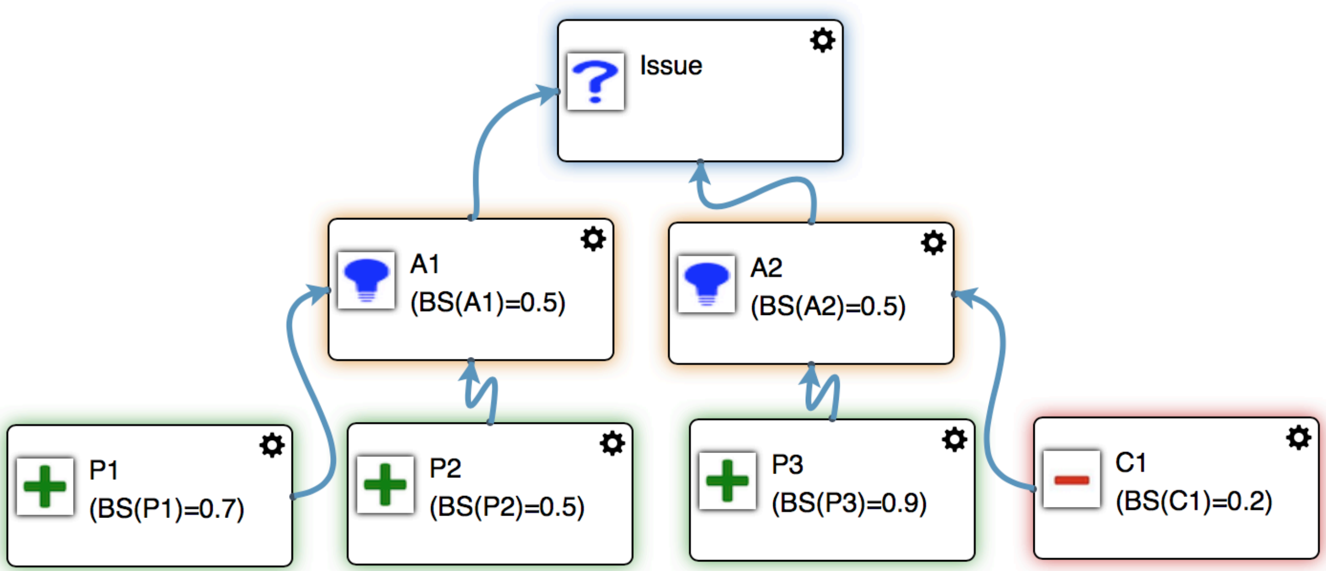
- Engineering Design setting as used for QuAD.

QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.675

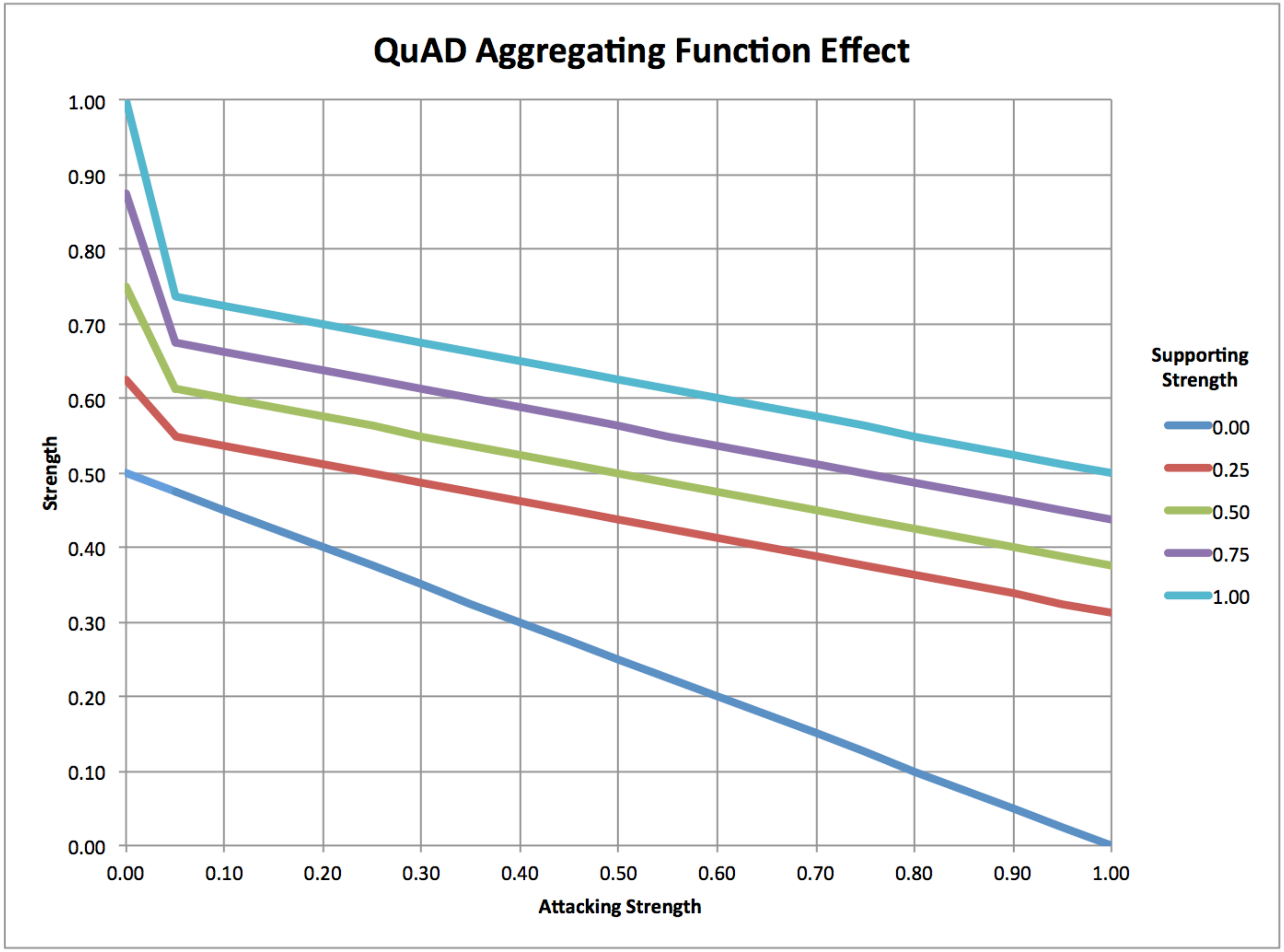
DF-QuAD Algorithm:

Answer	Strength at Stage 1	Strength at Stage 2
A1	0.925	0.925
A2	0.950	0.850



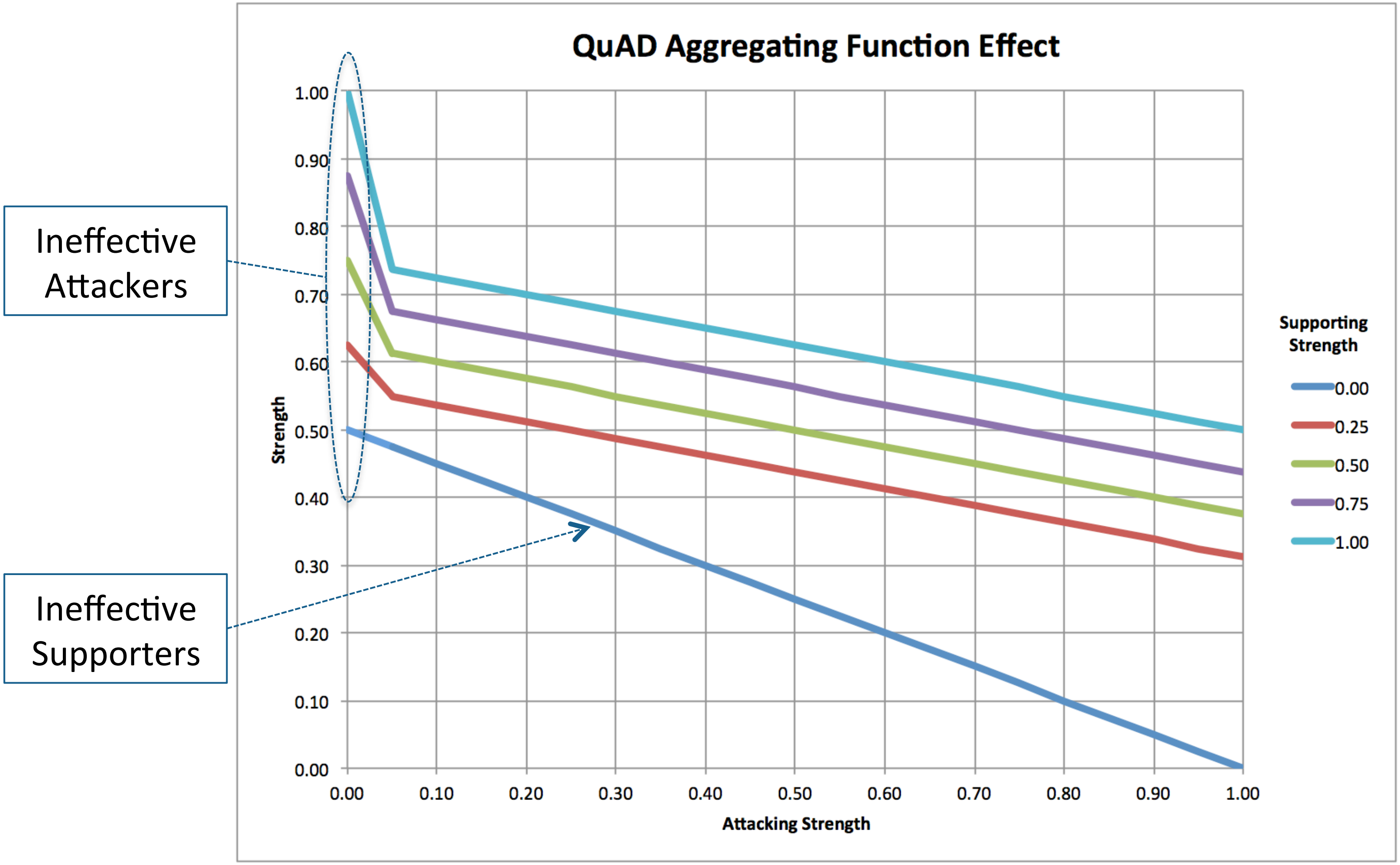
Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

- When QuAD's strength is plotted for a constant base score (0.5).



Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

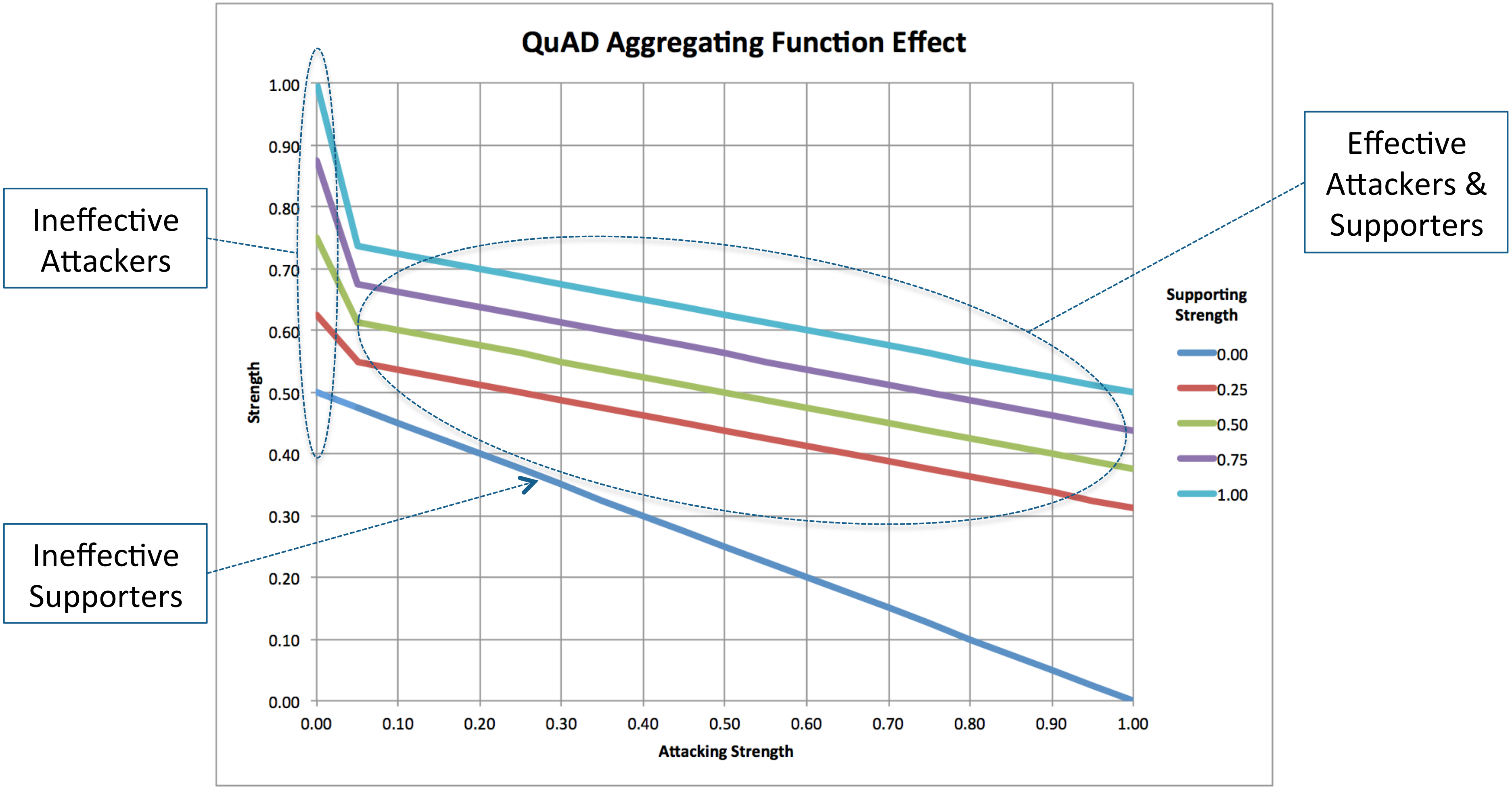
- When QuAD's strength is plotted for a constant base score (0.5).



Imperial College London

Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

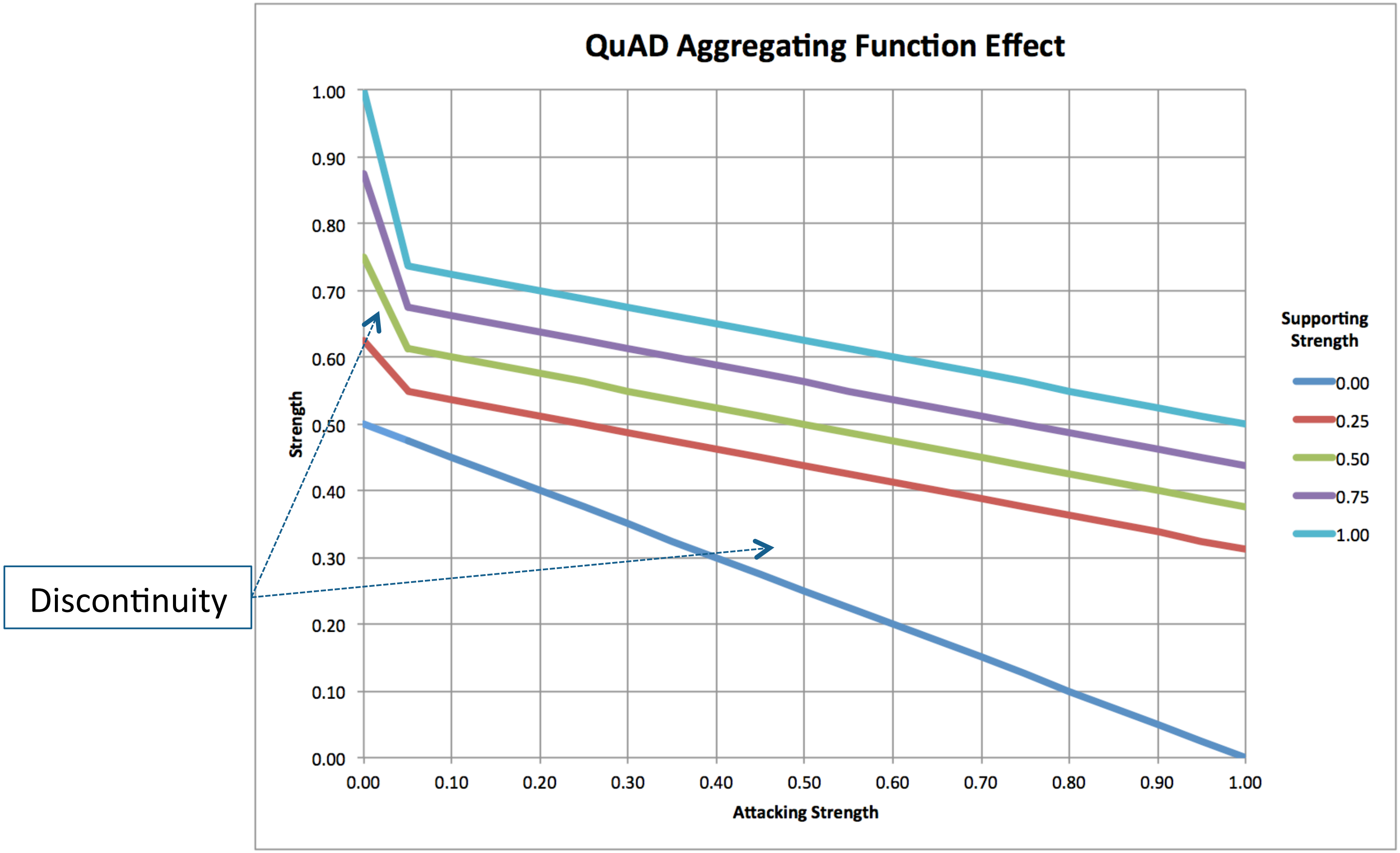
- When QuAD's strength is plotted for a constant base score (0.5).



Imperial College London

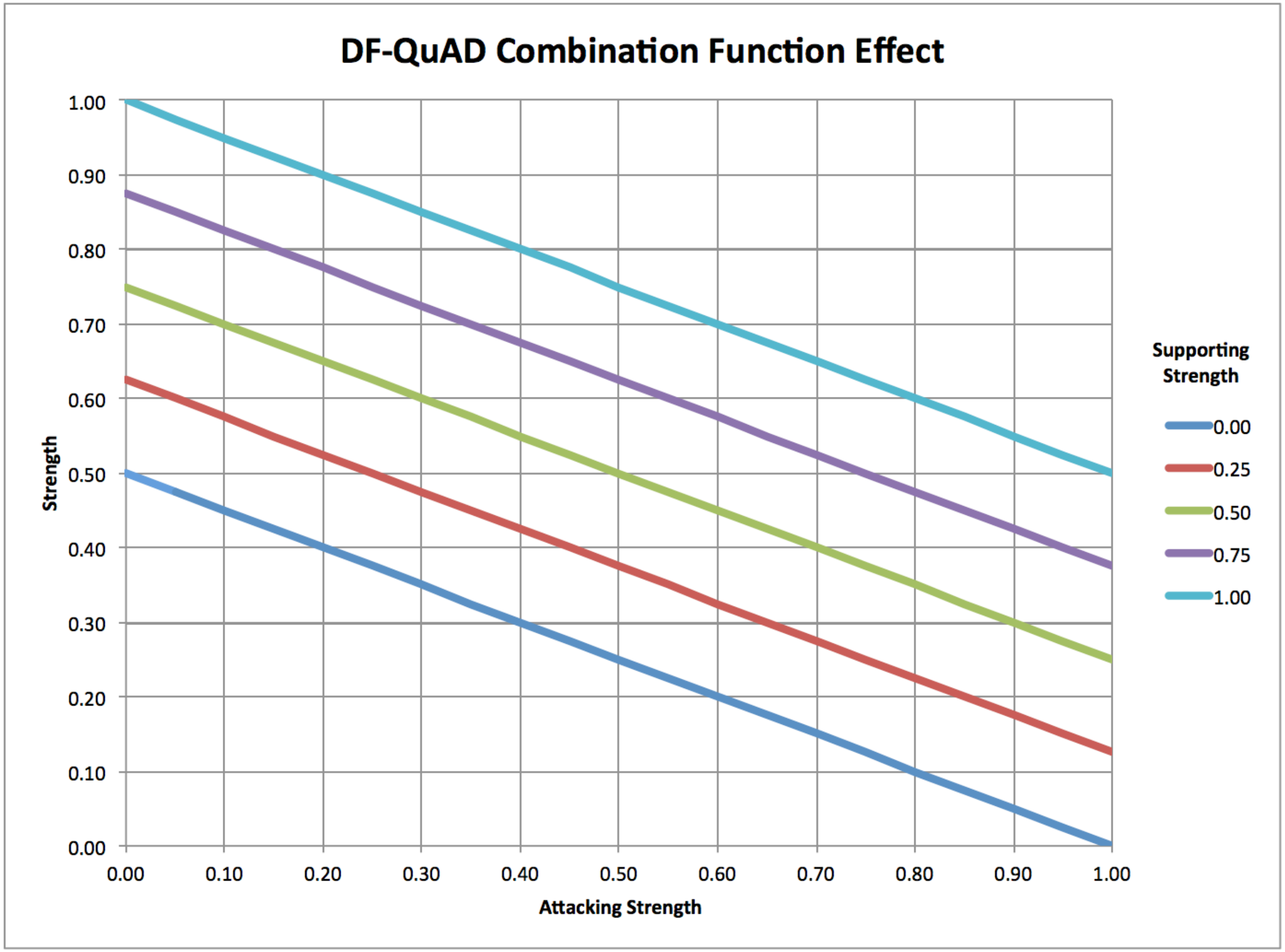
Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

- When QuAD's strength is plotted for a constant base score (0.5).



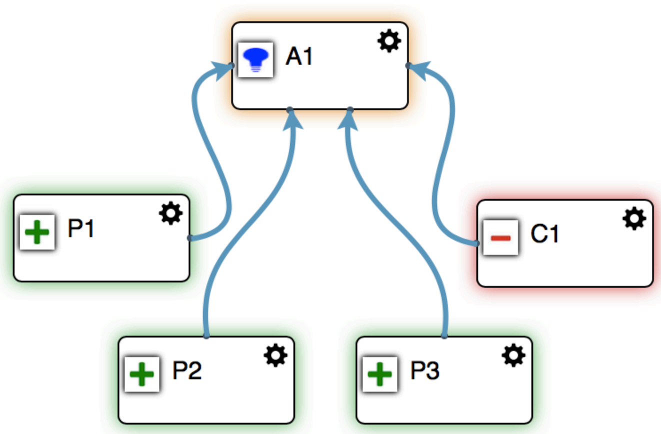
Research Summary – Comparison of the DF-QuAD and QuAD Algorithms

- DF-QuAD plotted for the same base score shows results without a discontinuity.



Research Summary – Properties Not Held by QuAD

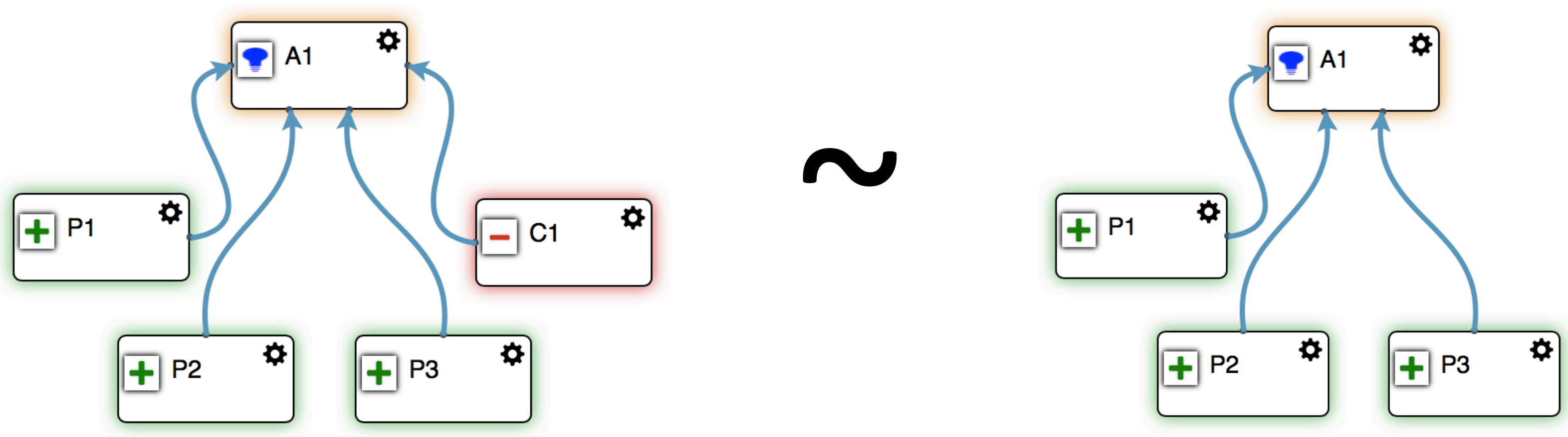
- As an attacker's or a supporter's strength approaches 0, the framework becomes equivalent to a framework without that argument:
  - As the strength of the attacker C1 approaches 0:



Imperial College London

Research Summary – Properties Not Held by QuAD

- As an attacker's or a supporter's strength approaches 0, the framework becomes equivalent to a framework without that argument:
  - As the strength of the attacker C1 approaches 0:

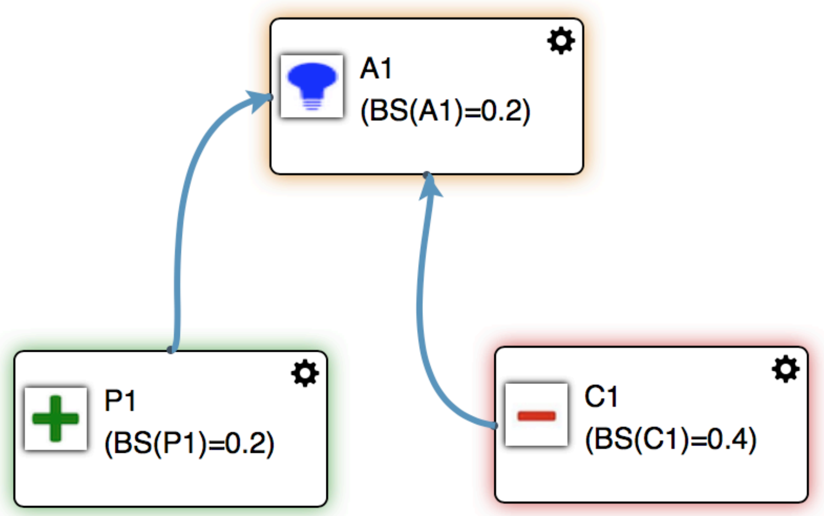


  - Similarly for supporters.
  - Gives DF-QuAD its discontinuity-free aspect.

Research Summary – Properties Not Held by QuAD

Condition	Strength
Argument's attacking component is larger than its supporting component	Less than or equal to the base score

- In DF-QuAD, the following results in a strength of A2 which is less than its base score:

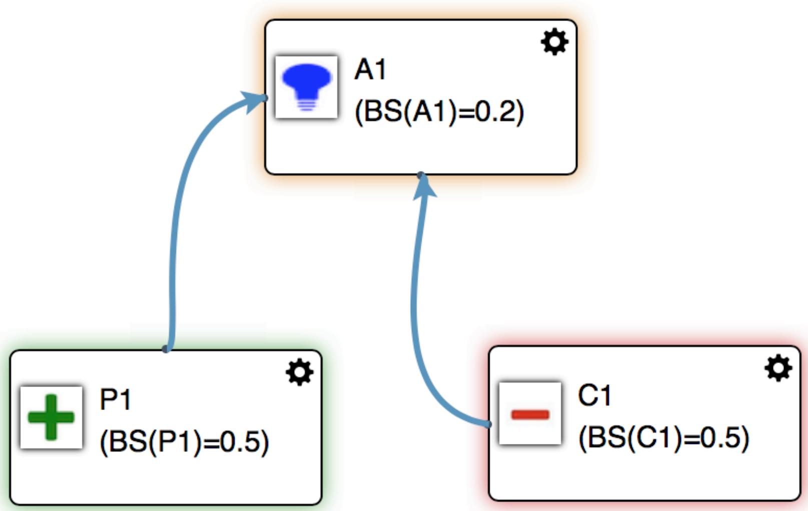


Answer	QuAD Strength	DF-QuAD Strength
A1	0.24	0.16

Research Summary – Properties Not Held by QuAD

Condition	Strength
Argument's attacking component is larger than its supporting component	Less than or equal to the base score
Argument's attacking component is equal to its supporting component	Equal to the base score

- In DF-QuAD, the following results in a strength of A2 which is equal to its base score:



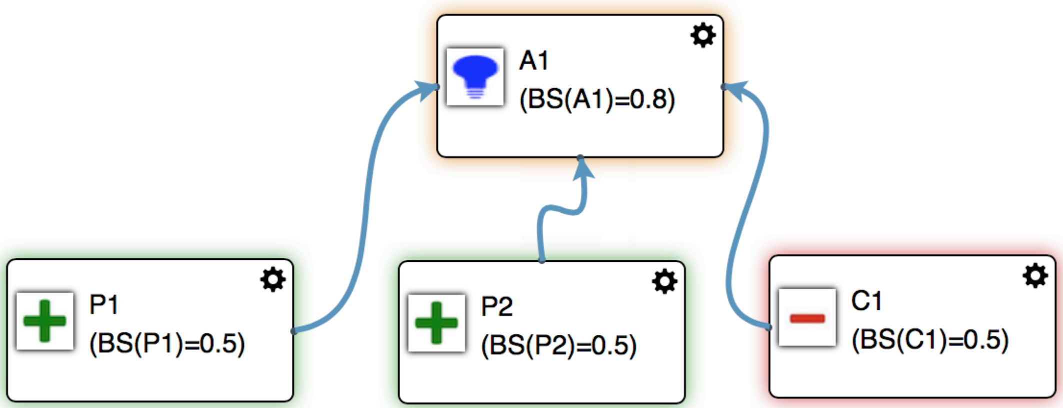
Answer	QuAD Strength	DF-QuAD Strength
A1	0.35	0.2

Imperial College London

Research Summary – Properties Not Held by QuAD

Condition	Strength
Argument's attacking component is larger than its supporting component	Less than or equal to the base score
Argument's attacking component is equal to its supporting component	Equal to the base score
Argument's attacking component is smaller than its supporting component	Greater than or equal to the base score

- In DF-QuAD, the following results in a strength of A2 which is greater than its base score:



Answer	QuAD Strength	DF-QuAD Strength
A1	0.675	0.85

Imperial College London

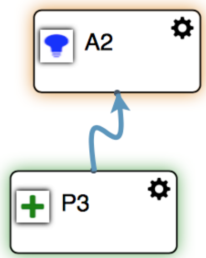
Research Summary – Properties Shared with QuAD

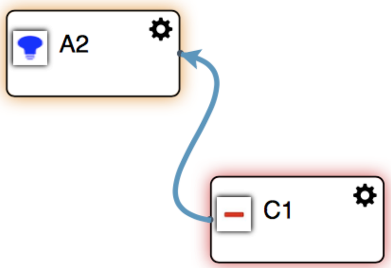
- The order of the sequence of attacking or supporting arguments does not affect the strength:

Pro before Con

Con before Pro

Stage 1





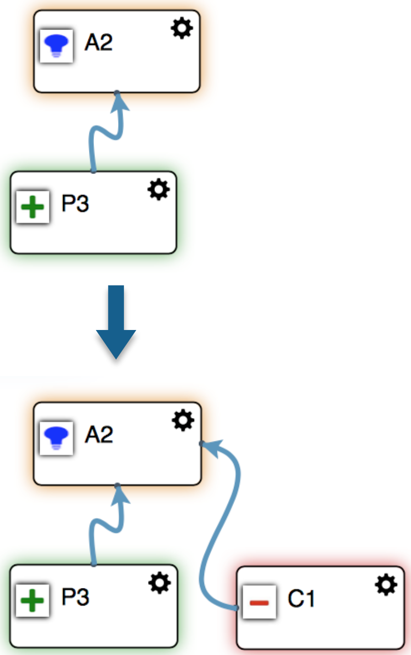
Research Summary – Properties Shared with QuAD

- The order of the sequence of attacking or supporting arguments does not affect the strength:

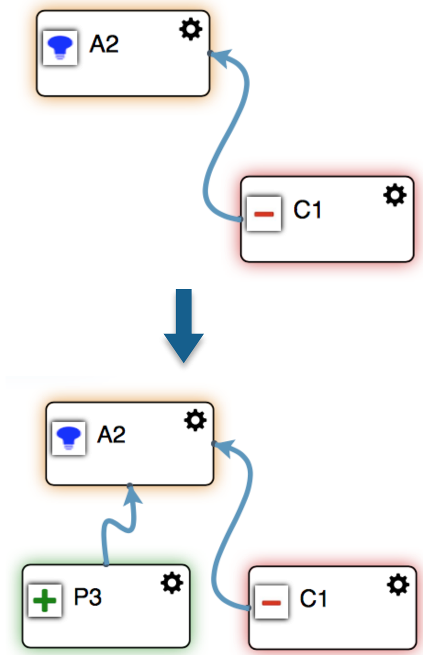
Stage 1

Stage 2

Pro before Con

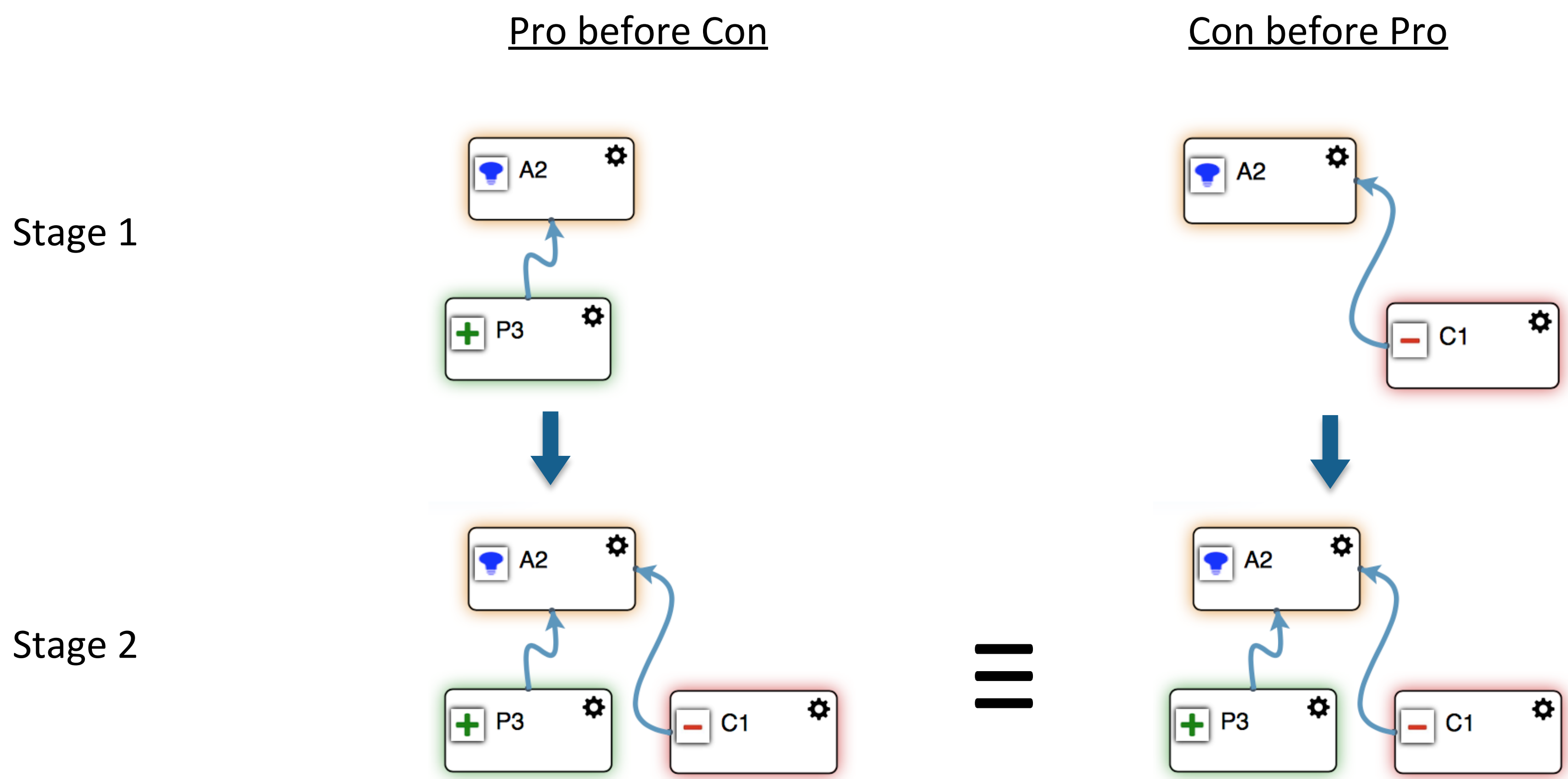


Con before Pro



Research Summary – Properties Shared with QuAD

- The order of the sequence of attacking or supporting arguments does not affect the strength:



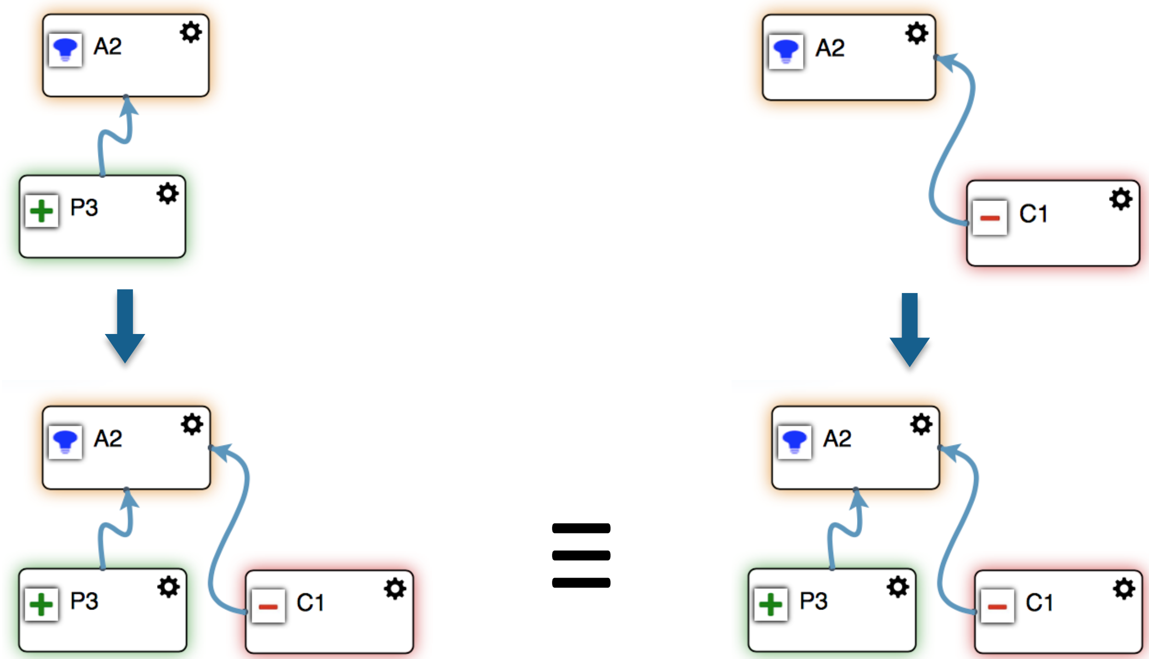
Research Summary – Properties Shared with QuAD

- The order of the sequence of attacking or supporting arguments does not affect the strength:

<u>Pro before Con</u> <u>Con before Pro</u>

Stage 1

Stage 2



- More intuitive, comprehensive properties also hold, for example:
  - An attacker being added will not increase the strength
  - A supporter being added will not decrease the strength

Research Summary – Relationship to Abstract Argumentation

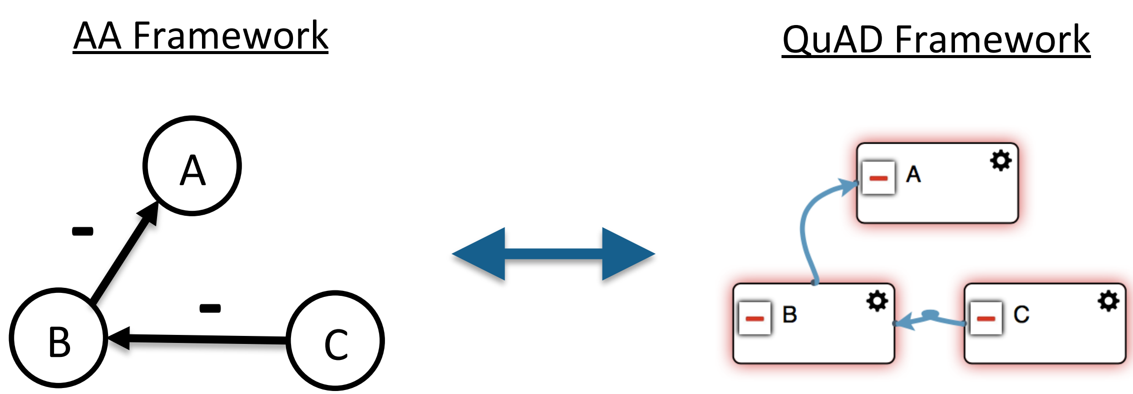
- QuAD and DF-QuAD yield the same results when attackers or supporters are ineffective.

- Abstract Argumentation frameworks [Dung, 1995] can be mapped to QuAD frameworks without supporters [Baroni et al. 2015].

Research Summary – Relationship to Abstract Argumentation

- QuAD and DF-QuAD yield the same results when attackers or supporters are ineffective.

- Abstract Argumentation frameworks [Dung, 1995] can be mapped to QuAD frameworks without supporters [Baroni et al. 2015].



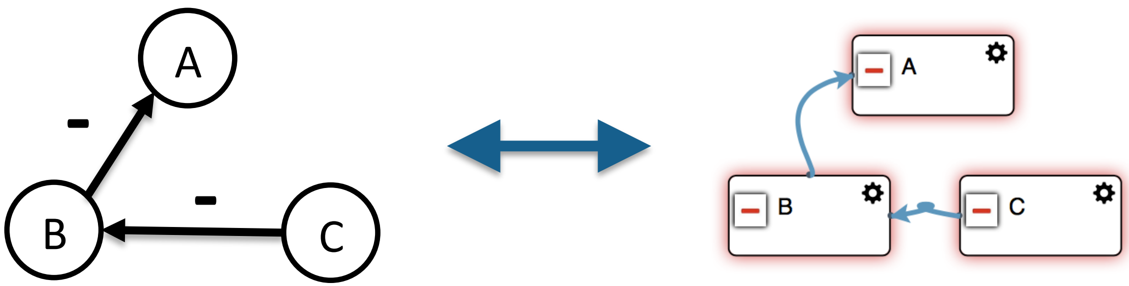
Research Summary – Relationship to Abstract Argumentation

- QuAD and DF-QuAD yield the same results when attackers or supporters are ineffective.

- Abstract Argumentation frameworks [Dung, 1995] can be mapped to QuAD frameworks without supporters [Baroni et al. 2015].

AA Framework

QuAD Framework



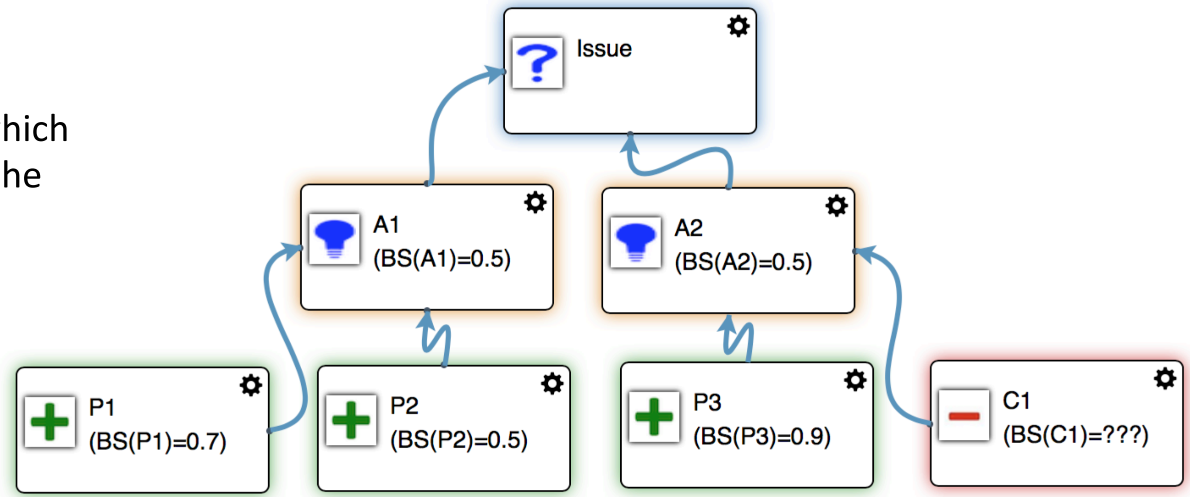
- If all base scores set to 1:
  - Arguments in grounded extension have strength of 1.
  - Other arguments have strength of 0.

Research Summary – Reverse Engineering Functionality

- Reverse engineering: a user can engineer the framework to give a required ranking or score.

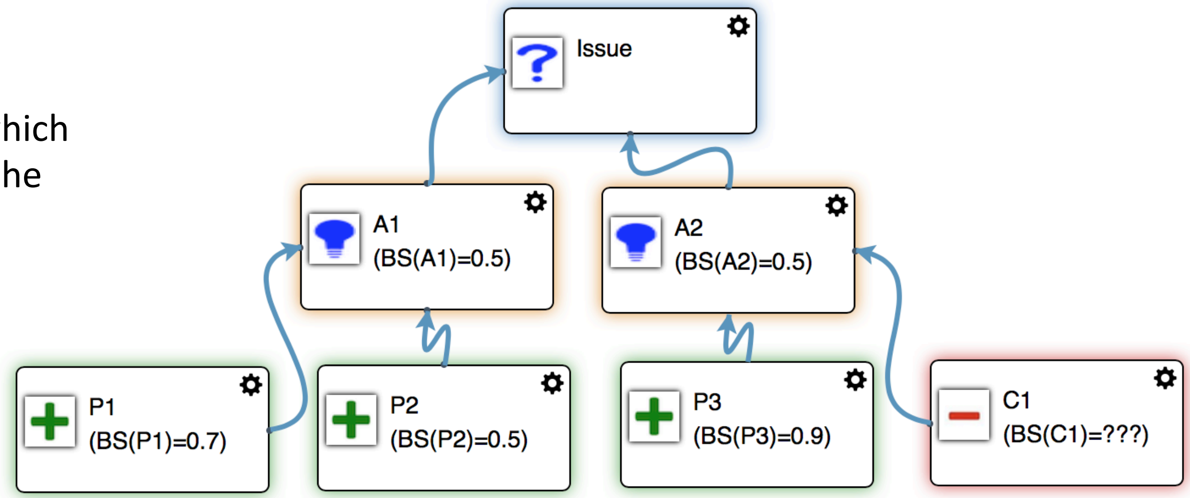
Research Summary – Reverse Engineering Functionality

- Reverse engineering: a user can engineer the framework to give a required ranking or score.
- Engineering Design Example:
  - Determine the BS(C1) at which the strength of A2 equals the strength of A1.



Research Summary – Reverse Engineering Functionality

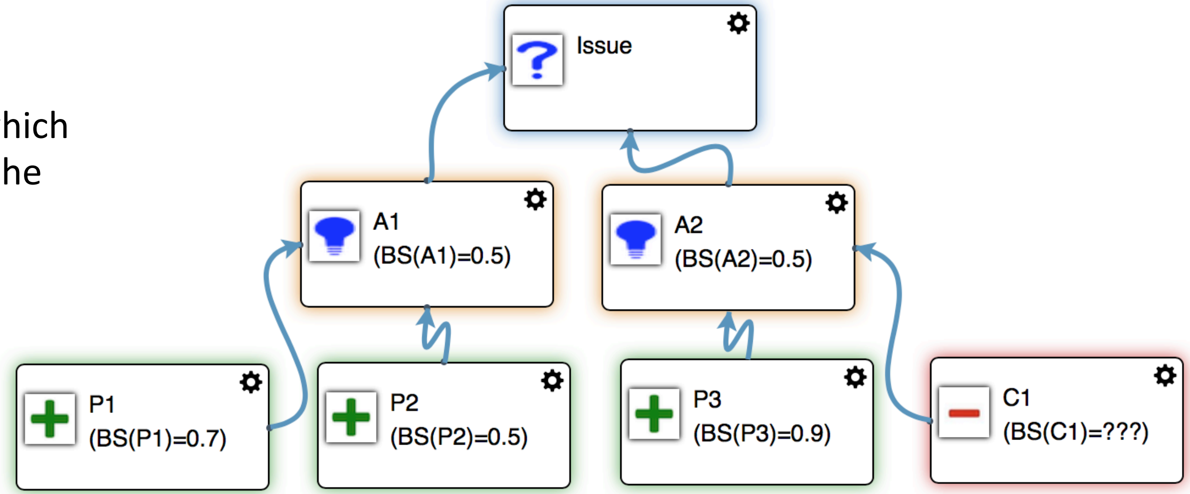
- Reverse engineering: a user can engineer the framework to give a required ranking or score.
- Engineering Design Example:
  - Determine the BS(C1) at which the strength of A2 equals the strength of A1.
  - BS(C1) = 0.05



Imperial College London

Research Summary – Reverse Engineering Functionality

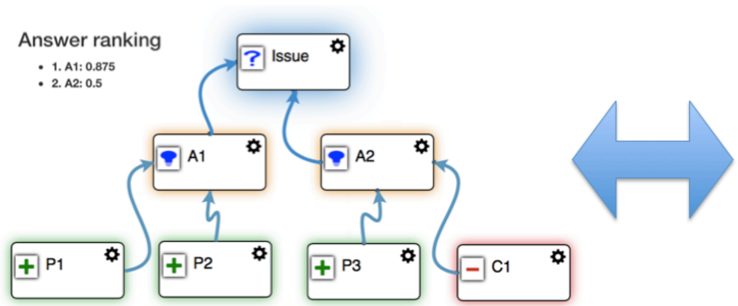
- Reverse engineering: a user can engineer the framework to give a required ranking or score.
- Engineering Design Example:
  - Determine the BS(C1) at which the strength of A2 equals the strength of A1.
  - BS(C1) = 0.05



- Other alterations include:
  - Increasing the strength of arguments by increasing base scores, e.g. increasing BS(P3) or BS(A2).
  - Reducing the strength of arguments by adding attackers, e.g. to C1.
  - Increasing the strength of arguments by adding supporters, e.g. to P3 or A2.
- Not possible in QuAD.
- Not desirable in some applications, e.g. where manipulation could be a problem.

Research Summary – Applications of QuAD Frameworks

- Arg&Dec [www.arganddec.com]
  - QuAD framework
  - QuAD automatic evaluation
  - Unique translation from graph to matrix form

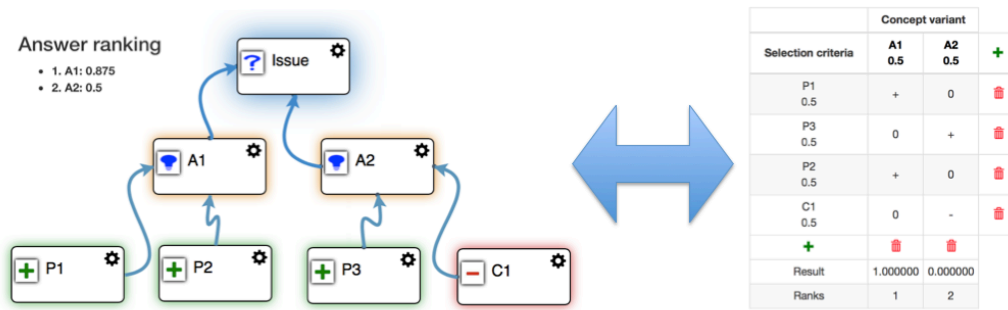


	Concept variant	
Selection criteria	A1 0.5	A2 0.5
P1 0.5	+	0
P3 0.5	0	+
P2 0.5	+	0
C1 0.5	0	-
Result	1.000000	0.000000
Ranks	1	2

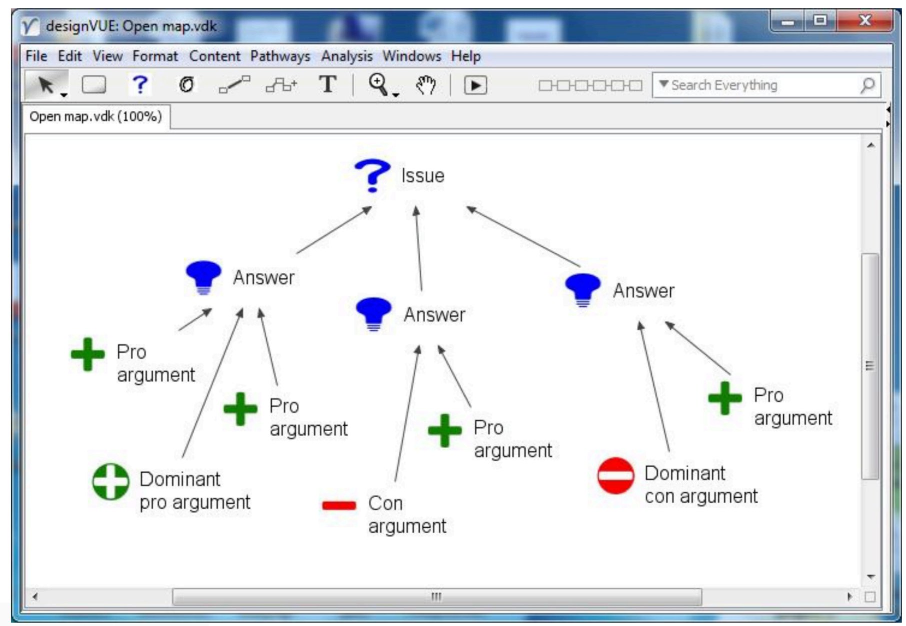
Imperial College London

Research Summary – Applications of QuAD Frameworks

- Arg&Dec [www.arganddec.com]
  - QuAD framework
  - QuAD automatic evaluation
  - Unique translation from graph to matrix form



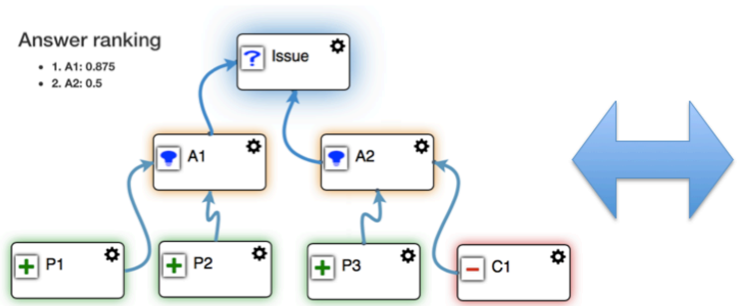
- designVUE [www3.imperial.ac.uk/designengineering/tools/designvue]
  - Debate mapping using IBIS graphs
  - Engineering design specific
  - Less limitations than Arg&Dec
  - Qualitative weighting measures



Imperial College London

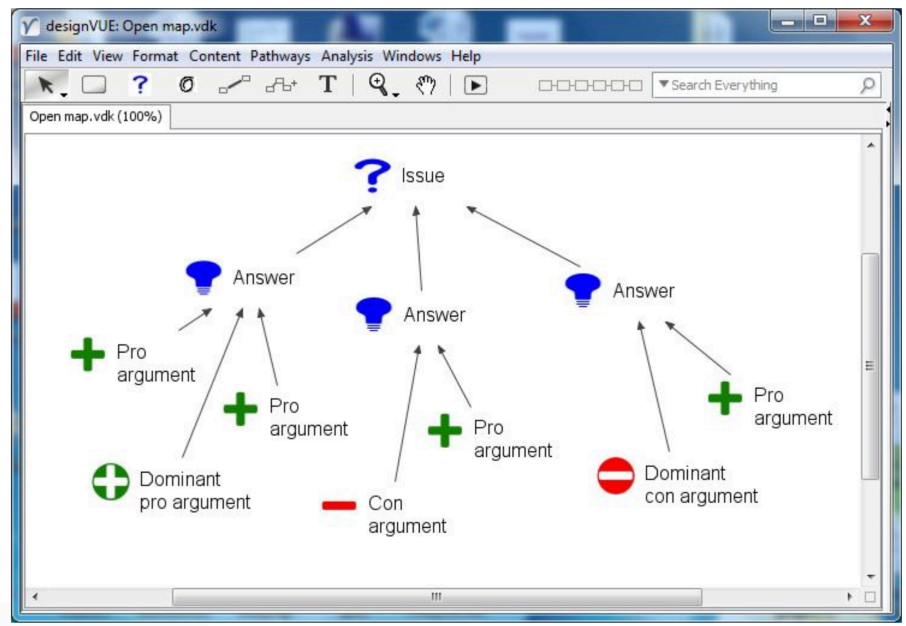
Research Summary – Applications of QuAD Frameworks

- Arg&Dec [www.arganddec.com]
  – QuAD framework
  – QuAD automatic evaluation
  – Unique translation from graph to matrix form



	Concept variant		
Selection criteria	A1 0.5	A2 0.5	+
P1 0.5	+	0	
P3 0.5	0	+	
P2 0.5	+	0	
C1 0.5	0	-	
+			
Result	1.000000	0.000000	
Ranks	1	2	

- designVUE [www3.imperial.ac.uk/designengineering/tools/designvue]
  – Debate mapping using IBIS charts
  – Engineering design specific
  – Less limitations than Arg&Dec
  – Qualitative weighting measures



- Other Applications
  – E-Democracy
  – Argument Mining
  – Medical Decision Support
  – Legal Reasoning

Future Work

- Further development of the QuAD framework:
  - Voting
  - Comparison with other algorithms/semantics for QuAD frameworks
  - Additional properties for algorithms [Amgoud & Ben-Naim, 2016]

Future Work

- Further development of the QuAD framework:
  - Voting
  - Comparison with other algorithms/semantics for QuAD frameworks
  - Additional properties for algorithms [Amgoud & Ben-Naim, 2016]
- Reverse engineering
  - Relationships to other alterations techniques, e.g. Enforcement [Bisquert & Cayrol, 2013]

Future Work

- Further development of the QuAD framework:
  - Voting
  - Comparison with other algorithms/semantics for QuAD frameworks
  - Additional properties for algorithms [Amgoud & Ben-Naim, 2016]
- Reverse engineering
  - Relationships to other alterations techniques, e.g. Enforcement [Bisquert & Cayrol, 2013]
- Relationships with other frameworks:
  - Matrix method [Aurisicchio et al. 2015]
  - Bipolar Argumentation Frameworks [Baroni et al. 2015; Amgoud et al. 2008]
  - Fuzzy Logic
  - Argumentation Labelling

Conclusions

- We have presented the DF-QuAD Algorithm:
  - Automatic evaluation within QuAD frameworks
  - Discontinuity-free algorithm
  - Shares many important properties with QuAD, and holds some new ones
  - Allows for reverse engineering

Thank You

Any Questions?