A Mechanism for Reasoning over Defeasible Preferences in Arg2P

Giuseppe Pisano[*] Roberta Calegari[°]
Andrea Omicini[⋆] Giovanni Sartor[†]

[*°†] Alma Mater Research Institute for Human-Centered Artificial Intelligence
[⋆] Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Italy
{g.pisano, roberta.calegari, andrea.omicini, giovanni.sartor}@unibo.it

CILC'21: 37th Italian Conference on Computational Logic
7-9 September 2021 - Parma, Italy

The CompuLaw project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 833647)

Context & motivation I

Argumentation with Conditional Preferences

- preferences depending on the admissibility status of an argument

→ many real life scenarios assume the ability tu argue over preferences

→ we often express preferences depending on the situation

? an efficient implementation of models for defeasible preferences is missing

→ we want to obtain the highest degree of *compatibility* with the standard approaches for *standard* and *structured* argumentation

Context & motivation II

Proposal

an efficient implementation for defeasible preferences in Arg2P *[Pisano et al., 2020, Calegari et al., 2020]*

→ the work is based on Dung's model for conditional priorities [Dung et al., 2019]

- provide an *optimized* implementation of the base model
- *generalise* the algorithm to handle *arbitrary* preference relations over arguments

Structured argumentation I

Defeasible rules

Rules that can be defeated by contrary evidence

$\rightarrow$ used to represent defeasible knowledge, i.e., tentative information that may be used if nothing could be posed against it

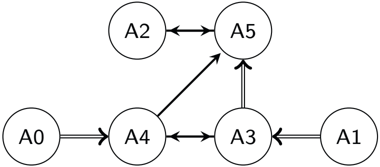
Standard argumentation theory [Modgil and Prakken, 2014]

- *defeasible theory* consists of a set of rules
- *arguments* can be constructed by chaining rules from the theory
- argument A *attacks* an argument B iff A undercuts, rebuts or undermines B

Structured argumentation II

Definition (Argumentation Framework (*AF*))

An **argumentation graph** constructed from a defeasible theory T is a tuple $\langle \mathcal{A}, \rightsquigarrow \rangle$, where $\mathcal{A}$ is the set of all arguments constructed from T, and $\rightsquigarrow$ is the *attack* relation over $\mathcal{A}$



Labelling semantics[Dung, 1995, Baroni et al., 2011]

Each argument is associated with one label which is either IN, OUT, or UNDmeaning the argument is either accepted, rejected, or undecided

Structured argumentation & Preferences I

Preference Arguments[Dung et al., 2019]

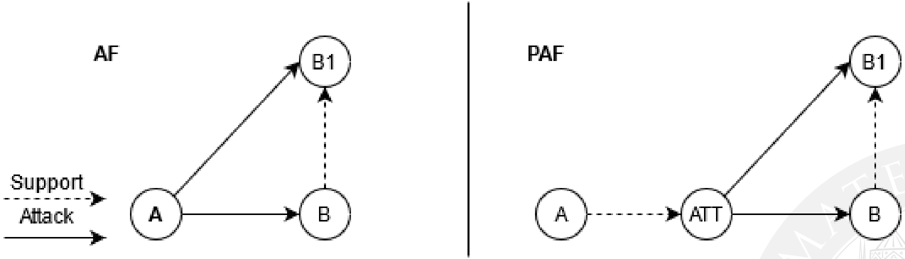
An argument such that its conclusion has the form $N(r1) \prec N(r2)$ where $r1$ and $r2$ are *defeasible rules*.

Preference based Argumentaion Framework (PAF)[Dung et al., 2019]

PAFs are a *transformation* of a standard AF also accounting for preference arguments.

- transform AF attacks into arguments
  - → conditional preferences attack these new arguments, thus challenging their effect
- rebuild the attacks set considering
  1. attacks coming from attacks transformed into arguments
  2. attacks involving preference arguments

Structured argumentation & Preferences II



Structured argumentation & Preferences III

PAF conservativeness

PAF is a conservative *generalisation* of standard AFs

$\rightarrow$ *standard semantics* can be applied to PAFs to determine arguments admissibility

Preference attack relation

the original work by [Dung et al., 2019] is focused on the relation used to identify the preference attacks

- $patt \subseteq \mathcal{A} \times \text{dbut}_{\rightsquigarrow}$ such that $(X, (A, B)) \in patt$ iff $\exists d \in \text{LastDefRules}(A)$ such that $d < \text{TopRule}(B) \in \text{Conc}(X)$

$\rightarrow$ an attack is not valid iff exists a single preference against it

PAF in Arg2P I

Arg-tuProlog

- lightweight implementation of structured argumentation
- interoperability, portability, modularity, customisation

→ suitable for highly-distributed environments

  → query-mode evaluation

Optimizing the transformation

only attacks that are *impacted* by a preference argument are *transformed*

→ if there are no preference arguments the PAF is *equal* to the original AF

→ no computational overhead (keep the *graph size* small)

PAF in Arg2P II

AF to PAF transormation

```
(PafArguments, PafAttacks) BuildPafArgumentationGraph(Arguments, Attacks):
    (ValidAttacks, InvalidAttacks) = filterSupRelatedAttacks(Attacks)

    (NewArguments, NewAttacks) = convertAttacks(InvalidAttacks)

    PrefAttacks = buildPrefAttacks(Arguments, NewArguments)

    PafArguments = append(Arguments, NewArguments)
    PafAttacks = append(ValidAttacks, NewAttacks, PrefAttacks)
    return (PafArguments, PafAttacks)
```

1. filter attacks that are in no circumstances impacted by preference arguments
2. convert the preference-related attacks into arguments
3. build the set of attacks coming from preference arguments
4. return the new arguments and attacks sets

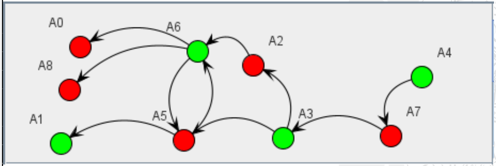
A simple example

Theory

```
r0 : [] => a.
r1 : a => b.
r3 : [] => -a.
r4 : [] => sup(r0, r3).
r5 : [] => sup(r3, r0).
r6 : [] => sup(r5, r4).
```

1. rules `r0` and `r1` concluding the two conflicting literals a and $\neg a$
2. `r1` claims b if a is proved
3. `r4`, `r5` and `r6` claim preferences over those rules

```
A0 : r0 ⟹ a
A1 : r3 ⟹ -a
A2 : r4 ⟹ sup(r0, r3)
A3 : r5 ⟹ sup(r3, r0)
A4 : r6 ⟹ sup(r5, r4)
A5 : A0,attack ⟹ attack(A0, A1)
A6 : A1,attack ⟹ attack(A1, A0)
A7 : A2,attack ⟹ attack(A2, A3)
A8 : A0,r1 ⟹ b
```



Going Further

Transformation Soundness

The transformation algorithm is sound w.r.t. the original model [Dung et al., 2019].

→ demonstration details in the paper

Should we stop here?

Our requirements demand to have *the highest degree of compatibility with the standard approaches of standard and structured argumentation*

→ What about the use of combined preferences?

The *ASPIC+* framework deals also with arguments relations based on *multiple* preferences (e.g. *weak democrat principle*)

→ We need to generalise standard *PAF* mechanisms

Generalising PAFs I

Grouped Preference Set & Joint preferences argument

Given a set of preference arguments $\mathcal{A}_P$ in an argumentation framework, the set of all the possible subsets of $\mathcal{A}_P$ ($2^{\mathcal{A}_P}$) is the **grouped preference set**

→ for every element of the grouped preference set we build an artificial argument called **joint preferences argument**.

From PAF to GPAF

1. the arguments set also contains the joint preferences arguments
2. preference attacks are not issued by single arguments but by joint preferences arguments
   → we can now have combination of rule preferences

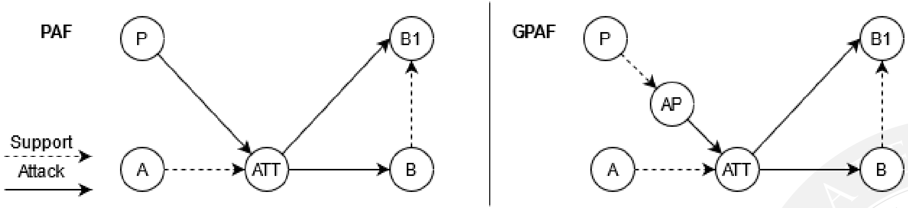
Generalising PAFs II

AF to GPAF transormation

```
(PafArguments, PafAttacks) BuildPafArgumentationGraph(Arguments, Attacks):
    (ValidAttacks, InvalidAttacks) = filterSupRelatedAttacks(Attacks)

    (NewArguments, NewAttacks) = convertAttacks(InvalidAttacks)

    PrefAttacks = buildPrefAttacks(Arguments, NewArguments)

    PafArguments = append(Arguments, NewArguments)
    PafAttacks = append(ValidAttacks, NewAttacks, PrefAttacks)
    return (PafArguments, PafAttacks)
```

1. filter attacks that are in no circumstances impacted by joint preferences arguments
2. convert the preference-related attacks into arguments
3. build the set of attacks coming from joint preferences arguments
4. return the new arguments and attacks sets

Generalising PAFs III



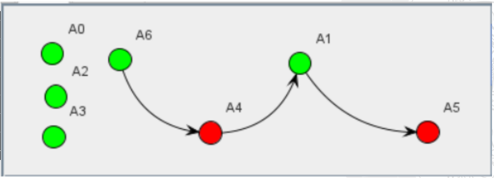
A simple example

Theory

```
r0 : [] => a.
r1 : a => b.
r2 : [] => -b.
r3 : [] => sup(r2, r1).
r4 : [] => sup(r2, r0).
```

1. rules r1 and r2 concluding the two conflicting literals b and $\neg b$
2. r1 claims b if a is proved
3. r4, r5 and r6 claim preferences over those rules
4. we apply the *weakest-link democratic ordering*

```
A0 : r0 ⟹ a
A1 : r2 ⟹ -b
A2 : r4 ⟹ sup(r2, r0)
A3 : r3 ⟹ sup(r2, r1)
A4 : A5,attack ⟹ attack(A5, A1)
A5 : A0,r1 ⟹ b
A6 : A2,A3,pref ⟹ mergedPreference
```



Future work

The work needs further analysis w.r.t. the themes of:

- computational complexity
  - *polynomial* w.r.t. the number of input attacks for both the algorithms
    - → $O(2N * PA)$ where N is the number of the input attacks and PA is the number of preference arguments
  - ! but a formal analysis is still missing
- termination
  - transformed graph should be *always* be obtainable by the proposed algorithms
  - *soundness* of solutions provided by the *transformed* argumentation graph must be proved

Also a *formal foundation* for the GPAF model needs to be provided

A Mechanism for Reasoning over Defeasible Preferences in Arg2P

Giuseppe Pisano[*] Roberta Calegari[○]
Andrea Omicini[⋆] Giovanni Sartor[†]

[*○†]Alma Mater Research Institute for Human-Centered Artificial Intelligence
[⋆]Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum—Università di Bologna, Italy
{g.pisano, roberta.calegari, andrea.omicini, giovanni.sartor}@unibo.it

CILC'21: 37th Italian Conference on Computational Logic
7-9 September 2021 - Parma, Italy

The CompuLaw project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 833647)

References I

[Baroni et al., 2011] Baroni, P., Caminada, M., and Giacomin, M. (2011).
An introduction to argumentation semantics.
The Knowledge Engineering Review, 26(4):365–410.

[Calegari et al., 2020] Calegari, R., Contissa, G., Pisano, G., Sartor, G., and Sartor, G. (2020).
Arg-tuProlog: a modular logic argumentation tool for PIL.
In Villata, S., Harašta, J., and Křemen, P., editors, *Legal Knowledge and Information Systems. JURIX 2020: The Thirty-third Annual Conference*, volume 334 of *Frontiers in Artificial Intelligence and Applications*, pages 265–268.

[Dung, 1995] Dung, P. M. (1995).
On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.
Artificial Intelligence, 77(2):321–358.

[Dung et al., 2019] Dung, P. M., Thang, P. M., and Son, T. C. (2019).
On structured argumentation with conditional preferences.
In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 2792–2800. AAAI Press.

References II

[Modgil and Prakken, 2014] Modgil, S. and Prakken, H. (2014).
The *ASPIC*$^{+}$ framework for structured argumentation: a tutorial.
Argument Computation, 5(1):31–62.

[Pisano et al., 2020] Pisano, G., Calegari, R., Omicini, A., and Sartor, G. (2020).
Arg-tuprolog: a tuprolog-based argumentation framework.
In *Proceedings of the 35th Italian Conference on Computational Logic - CILC 2020*, volume 2710 of *CEUR Workshop Proceedings*, pages 51–66. CEUR-WS.org.