# Towards cooperative argumentation for MAS: an Actor-based approach

Giuseppe Pisano* Roberta Calegari°
Andrea Omicini*

*°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}@unibo.it

WOA'21: 22nd Workshop "From Objects to Agents"
1-3 September 2021 - Bologna, 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

## Cooperative argumentation for MAS

- distributed and collaborative MAS (agents cooperate towards a goal)

→ conflict resolution

→ resolving conflicts by exploiting *negotiation via argumentation*

? well-founded computational model of argumentation is currently missing

→ difficult to investigate the *convergence* and *scalability* of argumentation techniques in *highly-distributed environments*

# Context & motivation II

## Proposal

*message-based distributed argumentation algorithm as the basic pillar of a computational model for cooperative argumentation in MAS*
→ actors paradigm and its main properties

- ignore issues such as autonomy and MAS coordination artefacts
- focus on distribution issues of cooperative argumentation based on the logic-based agreement framework Arg2P [Pisano et al., 2020, Calegari et al., 2020]
  → enables agent dialogue and defeasible reasoning in MAS

# Structured argumentation I

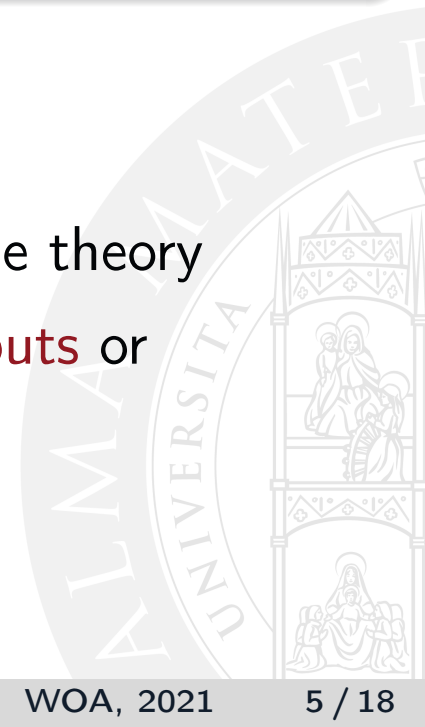
## Defeasible rules

Rules that can be defeated by contrary evidence

→ 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]

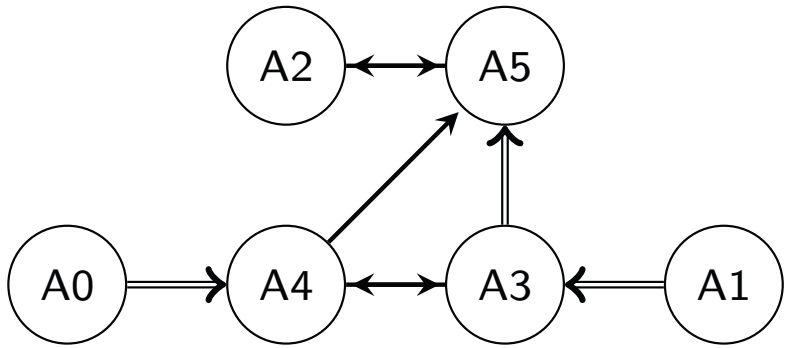
- *defeasible theory* consists of a set of rules
- *arguments* can be constructed by chaining rules from the theory
- argument $A$ *attacks* an argument $B$ iff $A$ undercuts, rebuts or undermines $B$



# Structured argumentation II

## Definition (Argumentation graph)

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

# Argument evaluation in Arg2P

## Arg2P

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

→ suitable for highly-distributed environments → query-mode evaluation

```
AnswerQuery(Goal):
 A1, ..., An = sustainingArguments(Goal)
 Res = ∅
 for A in A1, ..., An:
 Res = Res ∪ Evaluate(A, ∅)
 return Res.

Evaluate(A, Chain):
 if(∃ B ∈ Attacker(A):
   Evaluate(B, A ∪ Chain) = IN)
 return OUT
 if(∃ B ∈ Attacker(A): B ∈ Chain)
 return UND
 if(∃ B ∈Attacker(A):
   Evaluate(B, A ∪ Chain) = UND)
 return UND
 return IN.
```

Argument evaluation:

1. if a conflicting IN argument exists $A$ is OUT
2. if a cycle in the route from the root to the leaves exists $A$ is UND
3. if a conflicting UND argument exists $A$ is UND

If none $A$ is IN

# Desiderata

## Goal

Achieve a message-based distributed version of the above argumentation algorithm $\rightarrow$ requirements for a sound distributed evaluation of the argumentation task

We exploit the actors' paradigm and its main properties:

- fully reactive computational nodes
- communication through message passing

# Parallelisation

Parallelisation needs to be tackled under two distinct perspective:

- the computational task of the argument evaluation
  - → looking for possible sub-tasks to be parallelised within the algorithm itself
- the data used by the algorithm
  - → looking for possible data partitioning on which the argumentation process can be run in parallel

# Task parallelisation

- search and evaluation of the `Attackers`
  - → every condition exploited by the algorithm (1, 2, and 3) to evaluate an argument requires one and only one attacker to match the constraint
  - → **OR** parallelisation: evaluate the arguments simultaneously under different branches, and the success in one of the branches would lead to the success of the entire search
- evaluate all the conditions simultaneously and consider the ordering (strict) only while providing the final labelling
  - → mixing **AND** and **OR** parallelisation

# Data parallelisation

- split the *logic theory* avoiding contradictions

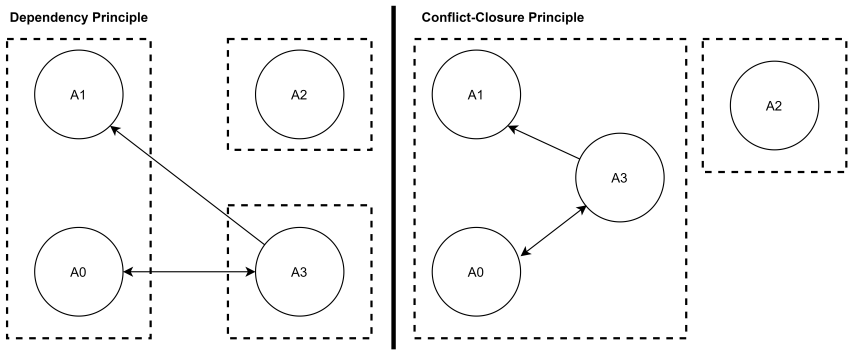
→ to guarantee a sound evaluation w.r.t. the original algorithm both *conflict-closure* and *dependency* principles should be considered

## Dependency principle

If two rules can be chained, they must stay together

## Conflict-closure principle

The graph-connected sub-portions maintain their integrity—i.e., attacker and attacked arguments grouped togheter



# Mixing task and data parallelisation

- A single subprocess in charge of evaluating an argument needs only the portion of the theory needed to infer the argument itself—i.e., the chainable rules concluding the target claim

→ search for attackers in argument evaluation is done by the a different subtask → attackers and attacked do not need to be together

→ only the *dependency* principle must be respected and the *conflict-closure* principle discarde.

# Theory distribution

## *Master/worker* approach

- master actors coordinate the knowledge base distribution phase
- workers hold a portion of the theory

→ masters' internal state contains a reference to the term to distribute and a list of the feedbacks from the workers' actors

→ workers' internal state simply represented by manage portion of theory

Add a new element to the theory (master action)

1. none of the workers contains a compatible knowledge
   - → the master creates a new worker containing the portion of the theory
2. one or more workers have a compatible knowledge base
   - → they add the element
3. overlapping knowledge in workers (possible creating unique inference chain)
   - → merge their knowledge and destroy the extra workers

# Argument evaluation I

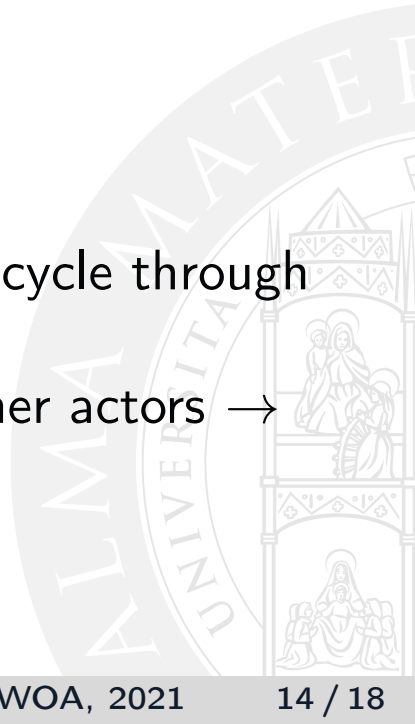
## Actor-based evaluation of an argument

→ `WorkerActor`

- each actor is responsible for evaluating those arguments that can be build using its portion of the theory

When the actor receives an evaluation request:

- checks if attackers exist, w.r.t. its knowledge
- then the actor can:
  - register the impossibility to evaluate the argument: if a cycle through the evaluation chain is detected
  - require the attacker arguments' evaluation to all the other actors → answer once collected the response



# Argument evaluation II

Conditions to match while evaluating an argument are the same as the original algorithm:

- if one counterargument is found admissible the argument is OUT
- if any number of actors votes for the argument undecidability with none voting for the rejection the argument is UND
- if all the actors agree that no counterarguments can be provided as acceptable the argument as IN

Actors provide their suggestion on the state of the requested argument according to all the labels of their counterarguments

# Future work

Our work can be extended in various directions:

- provide an implementation of the model in the Arg2P framework
- compare the performances of the monolithic and distributed versions of the algorithm properly addressing a discussion on efficiency and scalability issues
- provide a well-founded analysis of the model, taking into account its soundness

# Open questions

Experiments need to be run in a MAS context to answer the many open questions:

- How could agents benefit from this mechanism?
- How could the use of coordination media impact on the actual model?
- What would be the consequences of using the model in a context where the nodes possess an arbitrary knowledge?
- ...

# Towards cooperative argumentation for MAS: an Actor-based approach

Giuseppe Pisano* Roberta Calegari°
Andrea Omicini*

*°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}@unibo.it

WOA'21: 22nd Workshop "From Objects to Agents"
1-3 September 2021 - Bologna, 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

[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.

[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.