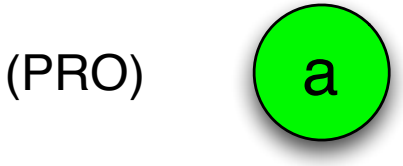
Imperial College Computing Student Workshop 2011
(from a previous TAFA11 paper and a future AAMAS12 submission)

Conditional Labelling for Abstract Argumentation

Guido Boella - University of Turin, Italy
Dov M. Gabbay - King's College London, UK
Alan Perotti - University of Turin, Italy
Leendert van der Torre - CSC, University of Luxembourg
Serena Villata - INRIA Sophia Antipolis, France

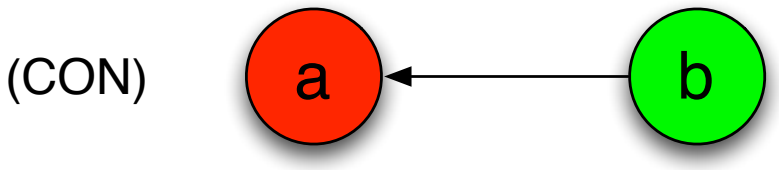
Argument games / Dialogues

- Agents debate about one (or more) topic(s), trying to defend it (proponents) or defeat it (opponents) by adding new arguments to the argumentation framework.
- Agents take turns in building a framework (no retraction).
- Easiest case: two agents (PRO,CON) argue about an argument *a*.



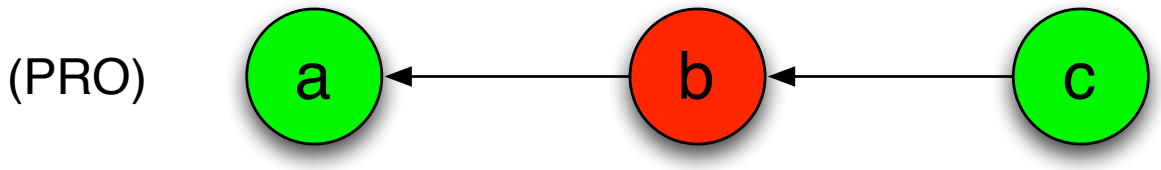
Argument games / Dialogues

- Agents debate about one (or more) topic(s), trying to defend it (proponents) or defeat it (opponents) by adding new arguments to the argumentation framework.
- Agents take turns in building a framework (no retraction).
- Easiest case: two agents (PRO,CON) argue about an argument *a*.



Argument games / Dialogues

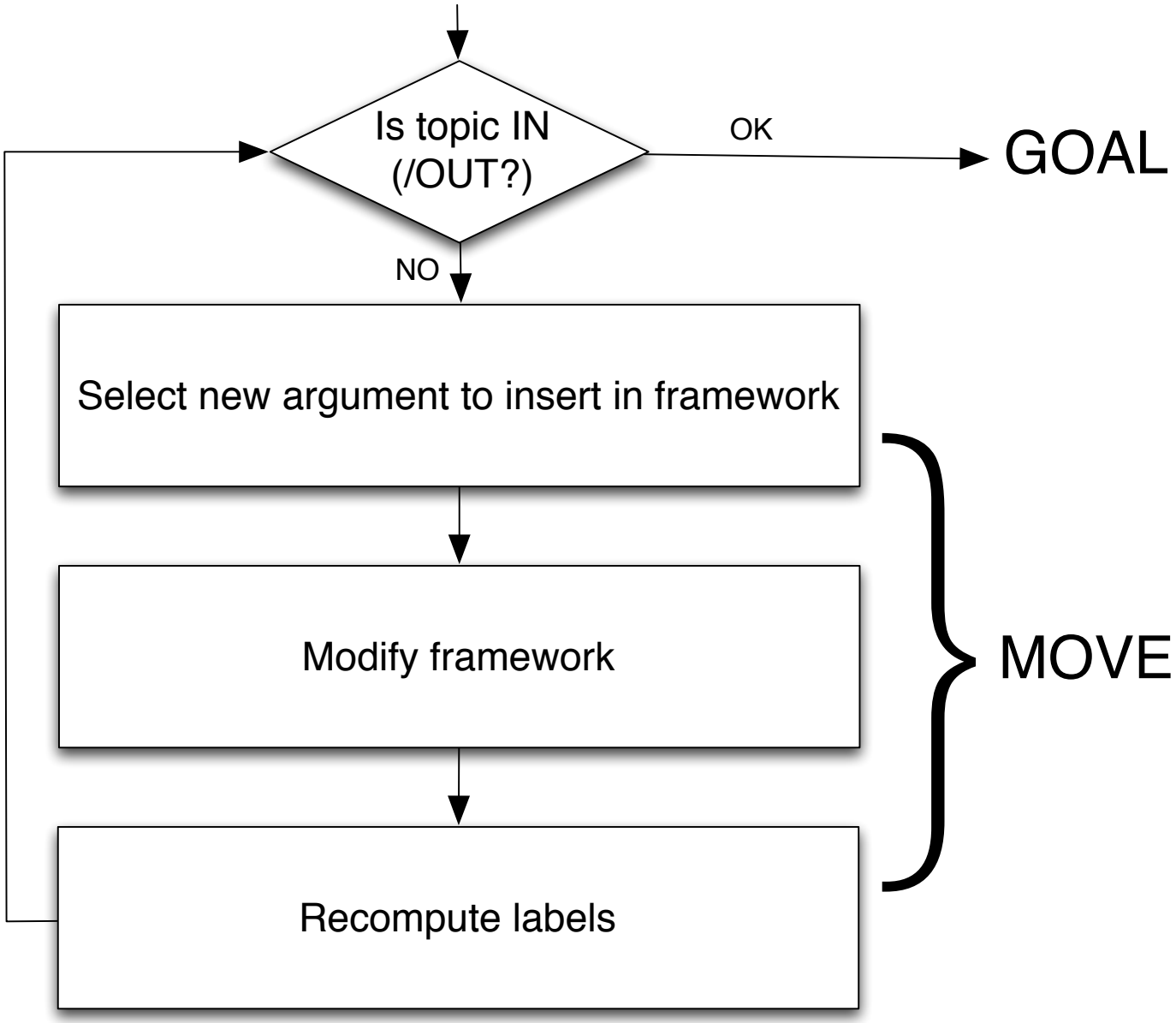
- Agents debate about one (or more) topic(s), trying to defend it (proponents) or defeat it (opponents) by adding new arguments to the argumentation framework.
- Agents take turns in building a framework (no retraction).
- Easiest case: two agents (PRO,CON) argue about an argument *a*.



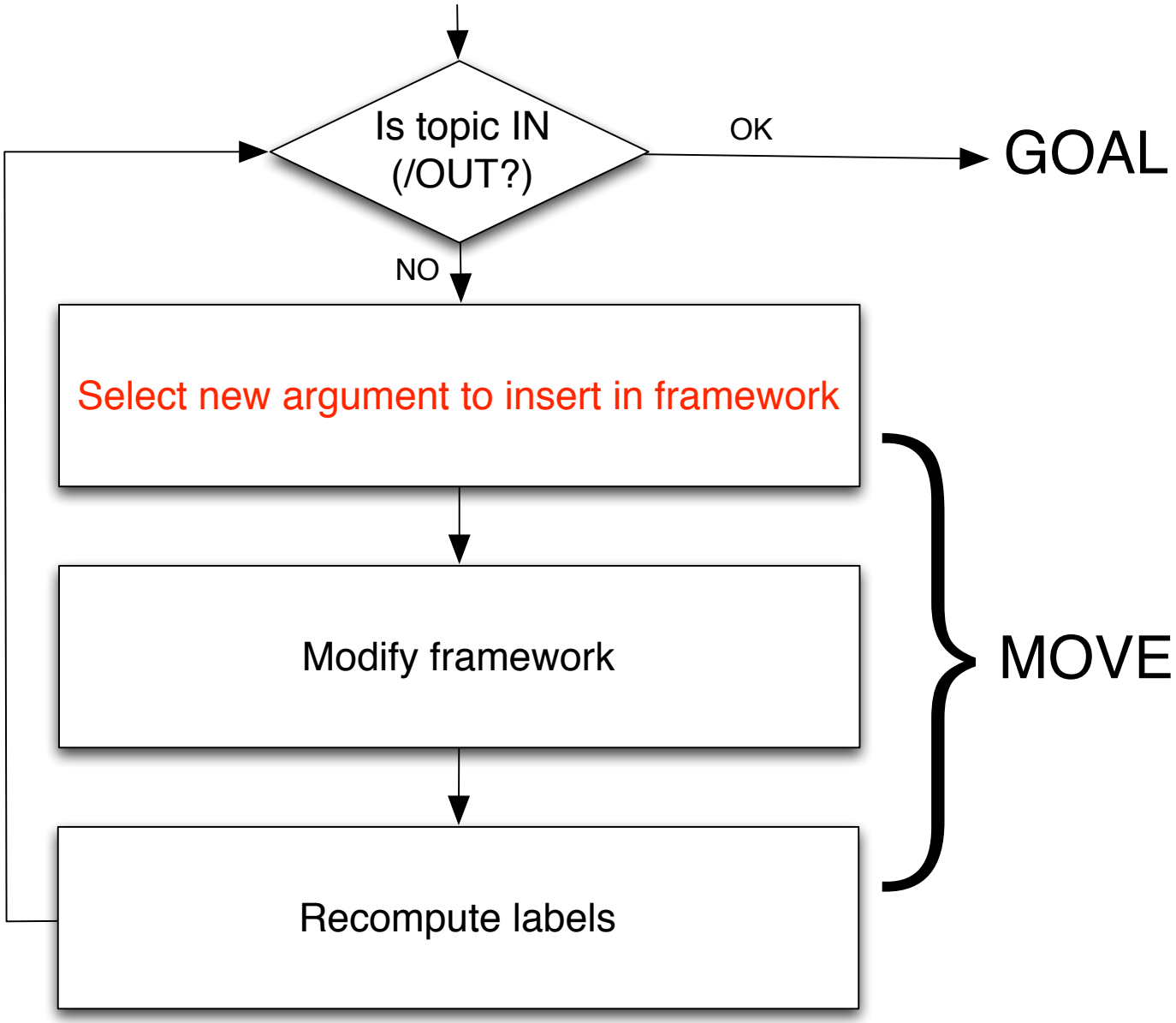
Moves and goals

- A MOVE is the insertion of a new argument into the framework.
- The GOAL is to have an argument accepted or rejected.

Moves and goals



Moves and goals



Research Question and Methodology

How to change an abstract argumentation framework, by introducing new arguments and their associated attacks, in order to have one or more arguments accepted or rejected?

Given an argumentation framework, we elicit information about the graph structure (such as paths and loops) and use them to compute information about **which argument should be defeated** in order to defend/defeat the argumentation topic

Research Question and Methodology

How to change an abstract argumentation framework, by introducing new arguments and their associated attacks, in order to have one or more arguments accepted or rejected?

Given an argumentation framework, we elicit information about the graph structure (such as paths and loops) and use them to compute information about **which argument should be defeated** in order to defend/defeat the argumentation topic

Research Question and Methodology

What kind of information can we associate to each argument concerning its possible justification statuses depending on the acceptability of other arguments in the framework?
How to compute this information in an efficient way?

For each argument, we associate **three conditional labels** to it, one for every possible **justification status**. Given an argument and a justification status, the conditional label tells which (other) arguments need to be attacked in order to give the argument the desired justification status.

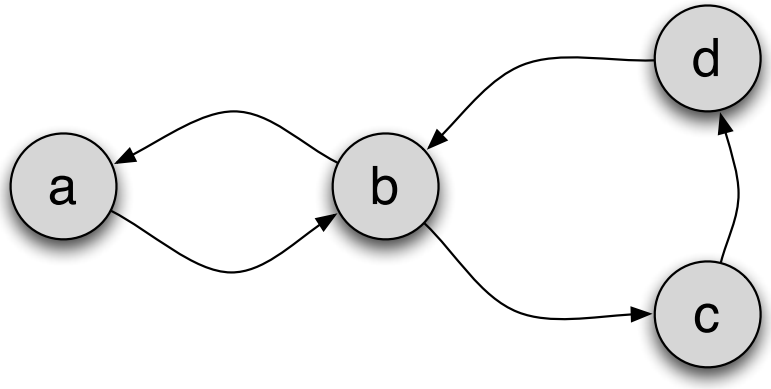
Research Question and Methodology

What kind of information can we associate to each argument concerning its possible justification statuses depending on the acceptability of other arguments in the framework?
How to compute this information in an efficient way?

For each argument, we associate **three conditional labels** to it, one for every possible **justification status**. Given an argument and a justification status, the conditional label tells which (other) arguments need to be attacked in order to give the argument the desired justification status.

Examples

Suppose that argument *a* is the topic and we want do defend it.
(that is, we want *a* to be labelled *in*.)

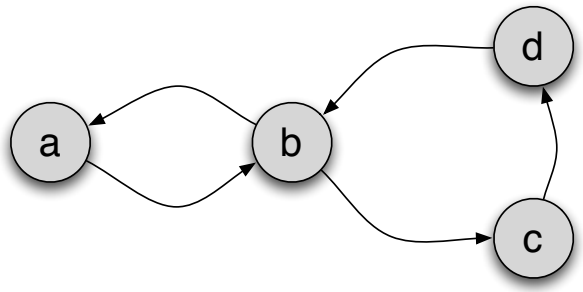


Should we attack *b*? *c*? Both of them? Is our goal even admissible?
What if we can't attack *b*? Can we still make an effective move?

Conditional labels

Given an argument a:

- a^+ means *make a in*
- a^- means *make a out*
- $a^?$ means *make a undec*
- a° means *defeat a*
- $\top$ means *success*
- $\bot$ means *failure*



$a^+ : b^\circ \wedge c^\circ \wedge d^\circ$

$a^- : a^\circ(\vee..)$

$a^? : \top$

$LABEL ::= HEAD : BODY$

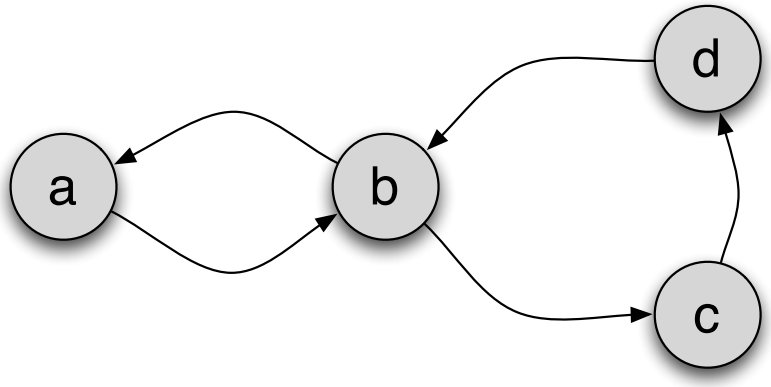
$HEAD ::= ARGNAME^+ \mid ARGNAME^- \mid ARGNAME^?$

$BODY ::= ARGNAME^\circ \mid \top \mid \bot \mid BODY \vee BODY \mid BODY \wedge BODY$

Approach: phases

1. associate each argument to three base labels (considering only the argument's direct attackers)
2. compute conditional labels by substitution
3. find target sets (for instance, by dnf-normalizing the formulae)
4. find a move such that it satisfies a target set of the goal formula.

1) Associate each argument to three base labels



[Caminada '06]

- An argument is *out* if it has an attacker which is *in*
- An argument is *in* if all of its attackers are *out*
- An argument is *undec* if it has at least one *undec* attacker and no *in* attacker

1) Associate each argument to three base labels

- $$a^+ = \bigwedge_{b \text{ s.t. } (b,a) \in R} b^-$$

  in order to ensure a's acceptance, all of a's attackers must be out.

- $$a^- = a^\circ \vee \bigvee_{b \text{ s.t. } (b,a) \in R} b^+$$

  in order to ensure a's rejection, either a is defeated or one of a's attacker is accepted.

- $$a^? = \left(\bigvee_{b \text{ s.t. } (b,a) \in R} b^? \right) \wedge \left(\bigwedge_{b \text{ s.t. } (b,a) \in R} b^- \vee b^? \right)$$

  in order to have an argument undecided, at least one of its attackers has to be undecided and all of them must be out or undecided.

2) Compute conditional labels by substitution

- $a^+ : b^? \wedge c^-$
- $b^? : c^- \vee \bot$

$\Rightarrow$

- $a^+ : (c^- \vee \bot) \wedge c^-$
- $b^? : c^- \vee \bot$

Basic substitution process

$$\frac{head_i : body_i \quad head_j : body_j \quad subf(head_j, body_i)}{head_i : body_i[body_j / head_j]}$$

2) Compute conditional labels by substitution

Simplification rules

- $\top \vee \alpha \rightsquigarrow \top$ (you either do nothing or do α: doing nothing is more convenient)
- $\bot \vee \alpha \rightsquigarrow \alpha$ (you can either fail or do α: in order to succeed you have to do α)
- $\top \wedge \alpha \rightsquigarrow \alpha$ (you have to both do nothing and α, therefore α)
- $\bot \wedge \alpha \rightsquigarrow \bot$ (you fail and you have to do α: you still fail)
- $\alpha \wedge \alpha \rightsquigarrow \alpha$
- $\alpha \vee \alpha \rightsquigarrow \alpha$
- $\alpha \vee (\alpha \wedge \beta) \rightsquigarrow \alpha$
- $\alpha \wedge (\alpha \vee \beta) \rightsquigarrow \alpha$

(compare to propositional connectives' properties)

2) Compute conditional labels by substitution

- $a^+ : b^- \vee a^-$
- $c^? : c^? \vee d^+$

$\Rightarrow$

- $a^+ : b^- \vee \bot$
- $c^? : \top \vee d^+$

Termination rules

$$\frac{head_i : body_i \quad head_i \equiv arg^{\pm} \quad subf(arg^{\pm}, body_i)}{head_i : body_i[\bot / arg^{\pm}]}$$

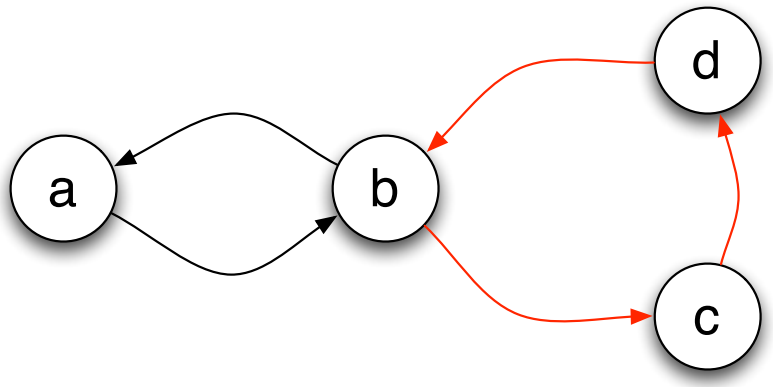
$$\frac{head_i : body_i \quad head_i \equiv arg^{?} \quad subf(arg^{?}, body_i)}{head_i : body_i[\top / arg^{?}]}$$

3,4) Find target sets and moves

- Start with a conditional label $(a^\circ \wedge (b^\circ \vee c^\circ))$
- Bring it to DNF $((a^\circ \wedge b^\circ) \vee (a^\circ \wedge c^\circ))$
- Get target sets $(\{\{a, b\},\ \{a, c\}\})$
- Find a move such that it modifies the framework according to a target set.

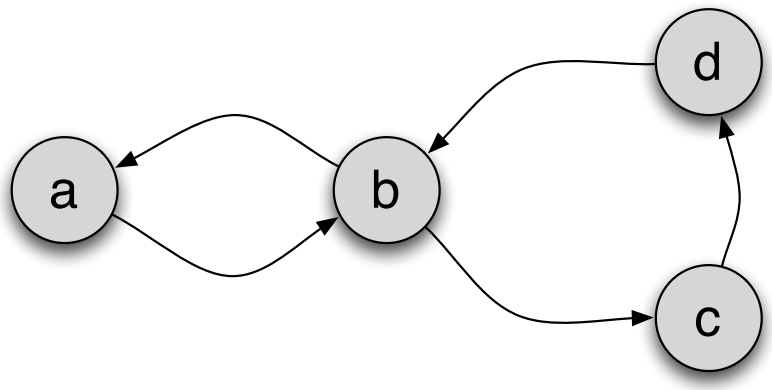
Optimization and loop detection

The biggest challenge lies in substitution, because the substitution process for each formula has the size of the framework as upper bound and the same substitutions take place several times, especially in highly connected frameworks.
A support for implementation can be a preprocessing phase of loop detection:



[Example] 1) Associate each argument to three base labels

Again, suppose that we want to defend a.



- $a^+ : b^-$
- $b^- : b^\circ \vee d^+$
- $d^+ : c^-$
- $c^- : c^\circ \vee b^+$
- $b^+ : a^- \wedge d^-$

for the sake of simplifications, only some labels are displayed.

[Example] 2) Compute conditional labels by substitution

- $a^+ : b^-$
- $b^- : b^\circ \vee d^+$
- $d^+ : c^-$
- $c^- : c^\circ \vee b^+$
- $b^+ : a^- \wedge d^-$

$a^+ : b^-$

$\rightsquigarrow \; a^+ : b^\circ \vee d^+$ [substitution]

$\rightsquigarrow \; a^+ : b^\circ \vee c^-$ [substitution]

$\rightsquigarrow \; a^+ : b^\circ \vee c^\circ \vee b^+$ [substitution]

$\rightsquigarrow \; a^+ : b^\circ \vee c^\circ \vee (a^- \wedge d^-)$ [substitution]

$\rightsquigarrow \; a^+ : b^\circ \vee c^\circ \vee (\bot \wedge d^-)$ [termination]

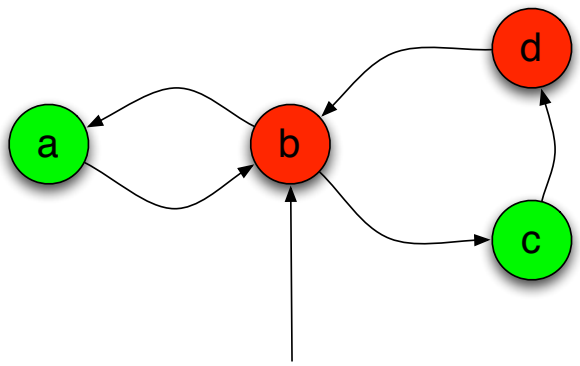
$\rightsquigarrow \; a^+ : b^\circ \vee c^\circ \vee \bot$ [simplification]

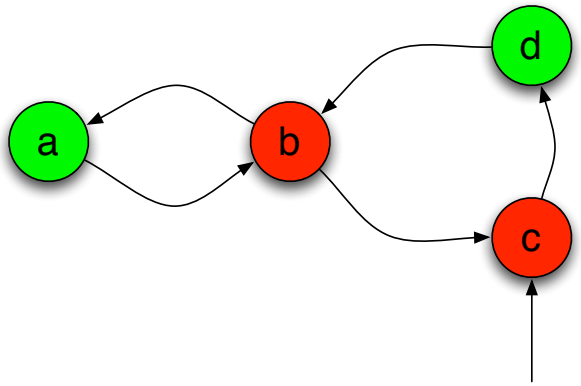
$\rightsquigarrow \; a^+ : b^\circ \vee c^\circ$ [simplification]

[Example] 3,4) Target sets and moves

DNF : $b^\circ \vee c^\circ$

Target Sets : $\{\{b\}, \{c\}\}$





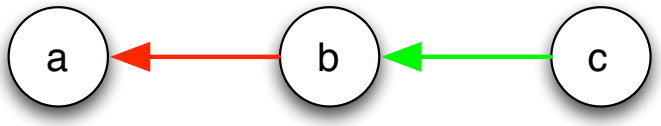
Summary

- New kind of labelling for abstract argumentation
- Plug-in for multiparty dialogues
- Link from desired justification state of a goal to possible moves
- Deal with graph topoi (loops, odd/even paths)
- Loop detection (adjacency matrix powers) as preprocessing

Ongoing work

[Boella, Villata, Van Der Torre](IJCAI 11)
Attack Semantics for Abstract Argumentation

- We shift our focus onto *arc successfulness*



[Bonzon, Maudet](AAMAS 11)
On the Outcomes of Multiparty Persuasion

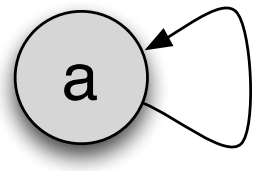
- We adopt their framework and extend their protocol
- We embed control (team's ability to back up)
- We use conditional labelling as a strategic, coalitional tool

References

- Phan Minh Dung: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif. Intell. (AI) 77(2):321-358 (1995).
- Martin Caminada: On the Issue of Reinstatement in Argumentation. JELIA 2006: 111-123.
- Guido Boella, Dov Gabbay, Alan Perotti, Leendert van der Torre, and Serena Villata: Conditional labelling for abstract argumentation, to appear In Procs. of the 1st Workshop on Theory and Applications of Formal Argumentation (TAFA), 2011.
- Elise Bonzon and Nicolas Maudet. On the Outcomes of Multiparty Persuasion. In Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2011), pp. 47–54, May 2011.
- Serena Villata, Guido Boella, Leon van der Torre: Attack Semantics for Abstract Argumentation, 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), IJCAI/AAAI, p. 406-413, 2011.

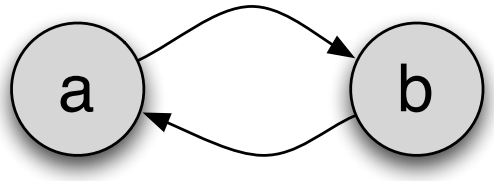
Thank you.

Example: single loop



- $a^+ : \bot$
- $a^- : a^\circ$
- $a^? : \top$

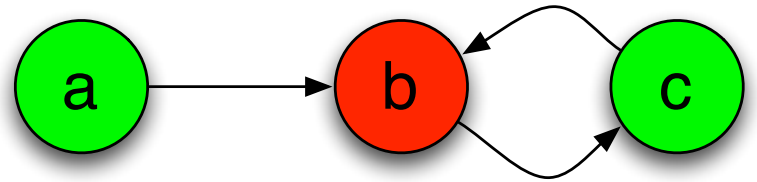
Example: 2-step loop



- $a^+ : b^\circ$
- $a^- : a^\circ$
- $a^? : \top$

- $b^+ : a^\circ$
- $b^- : b^\circ$
- $b^? : \top$

Example: 3-step loop

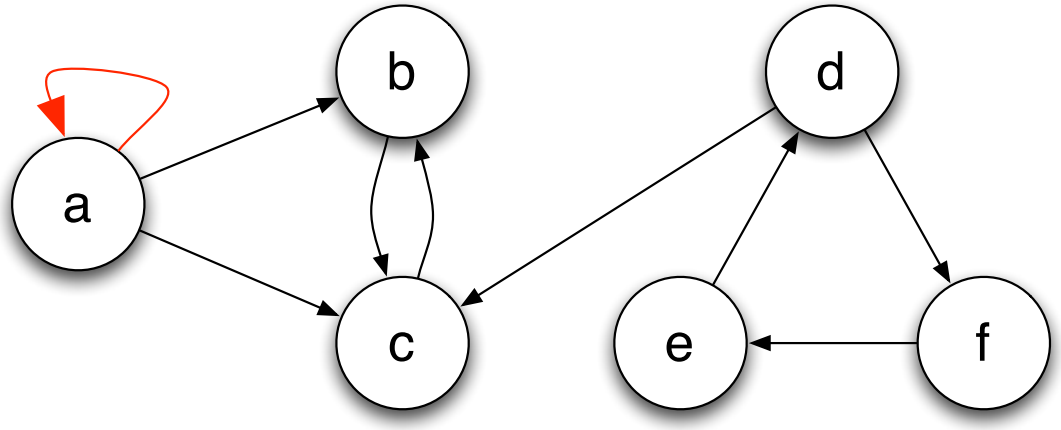


- $a^+ : \top$
- $a^- : a^\circ$
- $a^? : \bot$

- $b^+ : a^\circ \wedge c^\circ$
- $b^- : \top$
- $b^? : a^\circ$

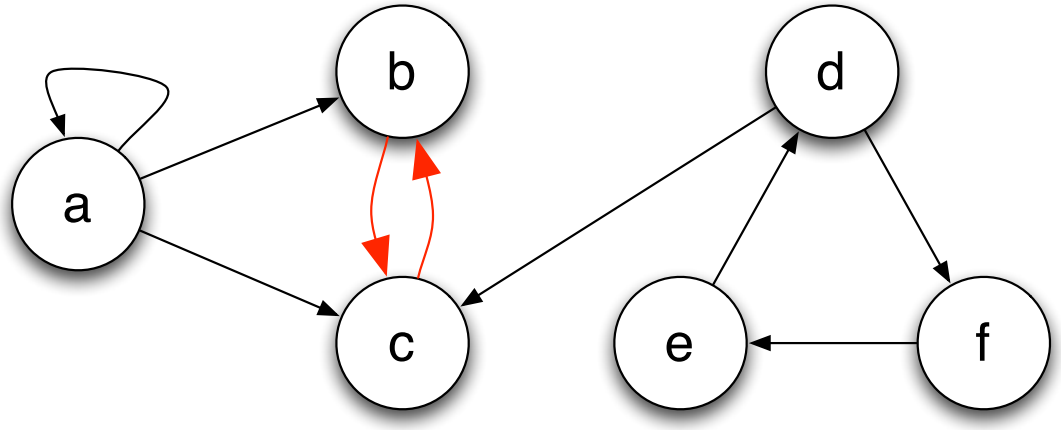
- $c^+ : \top$
- $c^- : c^\circ$
- $c^? : a^\circ$

Compute conditional labels by substitution



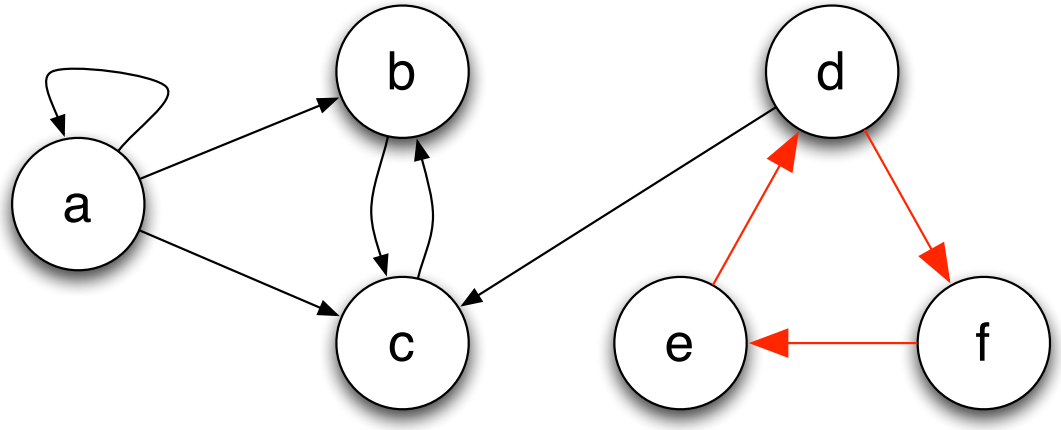
M^1	a	b	c	d	e	f
a	1	1	1	0	0	0
b	0	0	1	0	0	0
c	0	1	0	0	0	0
d	0	0	1	0	0	1
e	0	0	0	1	0	0
f	0	0	0	0	1	0

Compute conditional labels by substitution



M^2	a	b	c	d	e	f
a	1	1	1	0	0	0
b	0	1	0	0	0	0
c	0	0	1	0	0	0
d	0	1	0	0	1	0
e	0	0	1	0	0	1
f	0	0	0	1	0	0

Compute conditional labels by substitution



M^3	a	b	c	d	e	f
a	1	1	1	0	0	0
b	0	0	1	0	0	0
c	0	1	0	0	0	0
d	0	0	1	1	0	0
e	0	1	0	0	1	0
f	0	0	1	0	0	1

Language

$HEAD ::= FORM$
$HEAD ::= ARGNAME^{+} \mid ARGNAME^{-} \mid ARGNAME^{?}$
$FORM ::= ARGNAME^{\circ} \mid \top \mid \bot \mid FORM \vee FORM \mid FORM \wedge FORM$

- $a^{+} : \top$
  If your goal is to label a IN, you don't have to do anything
- $a^{-} : a^{\circ} \vee b^{+}$
  If you want to label a out, either attack it directly or get b labeled IN
- $a^{?} : \bot$
  you can't label a UNDEC