Conditional Labelling for Abstract Argumentation ⋆

Guido Boella[1], Dov M. Gabbay[2], Alan Perotti[1], Leendert van der Torre[3], and Serena Villata[4]

[1] Dipartimento di Informatica, Università di Torino {guido,perotti}@di.unito.it
[2] King's College London dov.gabbay@kcl.ac.uk
[3] ICR, University of Luxembourg leon.vandertorre@uni.lu
[4] INRIA, Sophia Antipolis serena.villata@inria.fr

Abstract. Agents engage in dialogues having as goals to make some arguments acceptable or unacceptable. To do so they may put forward arguments, adding them to the argumentation framework. Argumentation semantics can relate a change in the framework to the resulting extensions but it is not clear, given an argumentation framework and a desired acceptance state for a given set of arguments, which further arguments should be added in order to achieve those justification statuses. Our methodology, called *conditional labelling*, is based on argument labelling and assigns to each argument three propositional formulae. These formulae describe which arguments should be attacked by the agent in order to get a particular argument *in*, *out*, or *undecided*, respectively. Given a conditional labelling, the agents have a full knowledge about the consequences of their attacks on the acceptability of each arguments, without having to recompute the overall labelling of the framework for each possible set of attacks they may raise.

1 Introduction

Agents engage in dialogues having as goals to make some arguments acceptable or unacceptable: for instance, *agent A wins the auction* or *agent B is proven guilty*. At each turn, an agent owns a set of possible arguments she can add to the framework: each addition of further arguments to the framework is called a *move*. Argumentation semantics allow us to relate the introduction of a new argument (a *move*) to the resulting justification status of an argument (the *goal*): for instance, *if you defeat argument α then argument β will be labeled undec*. What is missing is a mechanism for making inferences from goals to moves: suppose an agent wants to make an argument β *undec*. How can she compute which arguments to add in order to achieve this goal? What she can do is to try and simulate the introduction of every possible argument she owns in the framework and then compute β's resulting label, comparing it to her goal. Beside this exhaustive approach there is no way, so far, for an agent to know which move to make in order to achieve her goal. Since reaching a goal may require the insertion

⋆ A longer version of this paper appeared in the proceedings of TAFA11 [2]

Andrew V. Jones (Ed.): Proceedings of the 1st Imperial College Student Workshop (ICCSW '11), pp. 59–65. September 29th–30th 2011, London UK.

of several arguments, the complexity of the exhaustive approach is exponential (wrt cardinality of the powerset) over the number of arguments an agent can add to the framework.

The research questions of the paper are:

1. 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?
2. How to compute this information in an efficient way?

We deal with abstract argumentation frameworks [4], where the internal structure of the arguments is left unspecified. We are inspired by Caminada's labelling [3], which assigns to each argument a label *in*, *out*, *undec*, and we extend this idea by assigning a triple of propositional formulae, called *conditional lables*, to every argument in the framework. These formulae are a guide in the dialogic process and suggest which move should be made next. Note that these formulae (and the algorithmic process to compute them) are in no way related to the number of agents: our approach does not depend on the number of argumenting agents and we apply it to a two-agent scenario for the sake of explanation. In this paper we focus on the *grounded* semantics, since it always allows to compute one single labelling. Our approach can be extended to deal with different semantics, but semantics with multiple or no extensions must be handled with care, in particular when investigating about credulous approaches to multiple extensions semantics.

2 Conditional labels

Our goal is to enrich each argument with some information about his *vulnerability*, i.e., we want to know how this argument could be successfully (even if indirectly) attacked, defended or made undecided. We purposely restrict our attention to argument defeating, due to two considerations: first of all, attacks are not resources but consequences of the insertion of the arguments and given a couple of arguments the existence of attacks between them is determined and not subject to strategic moves of agents. In second place, the building of an argumentation framework is a monotonic process and arguments can be defeated with new arguments rather than removed from the framework. Hence our proposal is to attach **three** formulae to each argument, meaning respectively

- Which arguments should be attacked in order to have this argument labelled *in*?
- Which arguments should be attacked in order to have this argument labelled *out*?
- Which arguments should be attacked in order to have this argument labelled *undec*?

Given an argumentation framework $\langle A, R \rangle$ (as defined in [4]), we associate to each argument α three formulae: α^+, α^-, $\alpha^?$. We indicate a generic formula associated to argument α as α^*. The language of the formulae is the same:

Definition 1. *(Language of conditional labels)*

- *if* $\beta \in A$, β° *is a formula.*
- $\top$ *and* $\bot$ *are formulae*
- *if* α_1^* *and* α_2^* *are formulae, also* $\alpha_1^* \wedge \alpha_2^*$ *and* $\alpha_1^* \vee \alpha_2^*$ *are.*

The interpretation of the formulae is: a formula α^+, if satisfied, guarantees that the related argument α is accepted (labelled *in*). The same holds for α^- formulae for *out* labels and $\alpha^?$ formulae for *undec* labels respectively. The atoms of those formulae are argument names β° or the special values $\top, \bot$.

- β° means *the agent has to defeat argument* β (to reach her goal)
- $\top$ means *the agent does not need to do anything* (to reach her goal)
- $\bot$ means *the agent can not do anything* (to reach her goal)

Due to space constraints, formal definition of the use of conditional labels is omitted; the main intuition will be just introduced informally. For techincal details see [2].

Each conditional label is composed by a head and a body; given an argument α, its three conditional labels are $\alpha^+ : body_\alpha^+$, $\alpha^- : body_\alpha^-$, $\alpha^? : body_\alpha^?$. A *targetset* for a label is a minimal set of arguments such that the arguments names are a solution for the label.

When we modify a framework via a move M we can defeat a set of arguments $defeat(M)$. If this set is one of the allowed target sets for the conditional label *lab* of an argument α, then the labelling of α in the resulting framework will be the one expressed by the head of the label *lab*: for instance, if $defeat(M)$ is a target set for $body_\alpha^+$, after M α's label will be *in* (same for $body_\alpha^-$ and *out* and $body_\alpha^?$ and *undec* respectively).

From a practical point of view, suppose that an agent wants to defend argument α: she has to compute the label α^+ and the target sets of the formula (that is, the minimal sets of solutions that satisfy the body of the label) are the arguments that have to be defeated in order to defend α.

3 Computing conditional labels

Our approach can be decomposed in four phases:

1. associate each argument to three base (or local) labels,
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.

The local labels correspond to:

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

The meaning of this formula is: *in order to ensure a's acceptance, all of a's attackers must be out.*

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

The meaning of this formula is: *in order to ensure a's rejection, either a is defeated or one of a's attackers is accepted.*

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

The meaning of this formula is: *in order to have an argument a undecided, at least one of a's attackers has to be undecided and all of a's attackers must be out or undecided.*
Note that this definition of grounded semantics mirrors Dung's original formulation ([4]).
The a° in the second formula means *a has to be defeated* and no substitution is required; b^+, b^- and $b^?$ refer to other formulae and have to be substituted to the actual formulae they refer to.

After this initial definition, the substitution process (phase two) takes place. It consists in substituting the references to other labels to those labels' actual values.

Simplifications need to be specified as follows:

- $\top \vee \alpha \rightsquigarrow \top$ (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)
- $\alpha \wedge \alpha \rightsquigarrow \alpha$
- $\alpha \vee \alpha \rightsquigarrow \alpha$
- $\alpha \vee (\alpha \wedge \beta) \rightsquigarrow \alpha$
- $\alpha \wedge (\alpha \vee \beta) \rightsquigarrow \alpha$

Let $i, j \in \{+, -, ?\}$. If α^i appears in the body of α^j:

- if $i = j = ?$, $\alpha^i \rightsquigarrow \top$
- else, $\alpha^i \rightsquigarrow \bot$

We express our termination conditions as simplification rules. The meaning is the following: if, substituting in the body of a conditional formula for an argument α, a conditional formula over the same argument is reached, the argument α belongs to a loop. So in this case the $a^?$ label is satisfied while a^+, a^- are not: if there is no way to give this argument an *in-out* label navigating the whole loop, it is pointless to go through the whole loop again.

4 Example

We now present some examples of conditional labelling.

Consider the example visualized in Figure 1.1. The basic labels are:

- $a^+ : b^-,\ \ a^- : b^+ \vee a^\circ,\ \ a^? : b^?$
- $b^+ : a^-,\ \ b^- : a^+ \vee b^\circ,\ \ b^? : a^?$

Solving the labels, for a we get $a^+ : b^\circ$, $a^- : a^\circ$, $a^? : \top$.

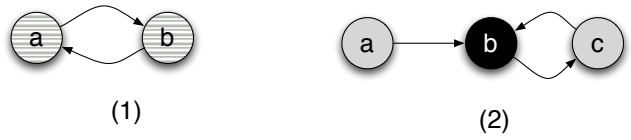

Fig. 1: Basic frameworks. Plain grey nodes represent *in* arguments and black nodes represent *out* arguments. *undec* arguments are depicted as dashed grey nodes.

Consider the example visualized in Figure 1.2. The basic labels are:

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

Consider argument b: it is *out*, but can be labelled *in* if we attack both a and c or *undec* if we attack a (thus activating the $b-c$ loop). We compute the conditional labels in the following way:

$b^+ : a^- \wedge c^-$
$= a \wedge (b^+ \vee c^\circ)$
$= a^\circ \wedge (\bot \vee c^\circ)$
$\rightsquigarrow a^\circ \wedge c^\circ$ (b can be labelled *in* by defeating a and c)

$b^- : a^+ \vee c^+ \vee b^\circ$
$= \top \vee b^- \vee b^\circ$
$= \top \vee \bot \vee b^\circ$
$\rightsquigarrow \top$ (no move is required in order to label b *out*)

$b^? : (a^? \vee c^?) \wedge (a^- \vee a^?) \wedge (c^- \vee c^?)$
$= (\bot \vee b^?) \wedge (a^\circ \vee \bot) \wedge ((b^+ \vee c^\circ) \vee b^?)$
$\rightsquigarrow (b^?) \wedge (a^\circ) \wedge ((b^+ \vee c^\circ) \vee b^?))$
$= (b^?) \wedge (a^\circ) \wedge ((\bot \vee c^\circ) \vee \top)$
$\rightsquigarrow (b^?) \wedge (a^\circ) \wedge (\top)$
$= (\top) \wedge (a^\circ) \wedge (\top)$
$\rightsquigarrow a^\circ$ (b can be labelled *undec* by defeating a)

The conditional labels computed mirror the intuitive description of the framework we gave and model it in a formal way.

5 Related Work

Conditional labelling is closely related to the dialogues games [5, 1]. Among others, Prakken [5] presents a formal framework for a class of argumentation dialogues, where each dialogue move either attacks or surrenders to a preceding move of the other participant. Amgoud and Hameurlain [1] argue that a strategy is a two steps decision process: i) to select the type of act to utter at a given step of a dialogue, and ii) to select the content which will accompany the act. Roth et al. [6] start from two principles: i) the outcome of a dispute depends on the strategies actually adopted by parties, but ii) this does not mean that the outcome can never be predicted because by using game theoretical solution concepts, the actions themselves can often be found. In comparison with this kind of frameworks, we share the idea that the first step consists in choosing the next move depending on the strategies of the agents. The differences are that we are not interested in providing the complete framework for argumentation dialogues games, we aim at providing a tool which can be used in those systems and which can be integrated with strategies. We do not restrict our framework to deal with two agents, and we extend the well-known argumentation labelling in order to provide a complete information about the argumentation framework on which it is applied.

6 Summary

In this paper we present a new kind of argument labelling, called conditional labelling. Conditional labelling allows to associate to each argument the information concerning its possible justification statuses, depending on the changes in the framework. In particular, we express this information by means of propositional formulae which express which arguments should be attacked in order to get the desired argument accepted, not accepted, or undecided. While it is quite straightforward to assign those conditional labels in argumentation frameworks without cycles and multiple attacks, it is rather complicated in the general case. When an argumentation framework with cycles is considered, it is possible to have in the conditional label α^* of an argument another α^* because the conditional labelling algorithm, using substitution, looks for all the attackers of the node until it finds the node itself. The conditional labelling allows the agents to avoid the exhaustive search of all the possible combinations in adding new arguments, and decreases the exponential complexity this search requires.

Future work addresses several issues: first of all, a deeper investigation on the complexity results related to the computation of the new labellings is necessary. From a purely argumentative perspective, we want to find out how conditional labels can be useful *after* a move: that is, if the previous information can be used to compute new conditional labels after the framework has been modified. Associating a cost concept to moves, our labelling lets agents link action costs to goals' outcomes, and can therefore be used as an underlying mechanism to develop strategies in a game theoretical context.

References

1. L. Amgoud and N. Hameurlain. A formal model for designing dialogue strategies. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *Autonomous Agents and Multiagent Systems (AAMAS)*, pages 414–416. ACM, 2006.
2. G. Boella, D. Gabbay, A. Perotti, L. van der Torre, and S. Villata. Conditional labelling for abstract argumentation. In *Procs. of the 1st Workshop on Theory and Applications of Formal Argumentation (TAFA), 2011.*
3. M. Caminada. On the issue of reinstatement in argumentation. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Journées Européennes sur la Logique en Intelligence Artificielle (JELIA)*, volume 4160 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2006.
4. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.
5. H. Prakken. Coherence and flexibility in dialogue games for argumentation. *J. Log. Comput.*, 15(6):1009–1040, 2005.
6. B. Roth, R. Riveret, A. Rotolo, and G. Governatori. Strategic argumentation: a game theoretical investigation. In *International Conference on AI and Law (ICAIL)*, pages 81–90. ACM, 2007.