Logical Modes of Attack in Argumentation Networks

Dov M. Gabbay
Department of Computer Science
King's College London
WC2R 2LS, London, UK
dg@dcs.kcl.ac.uk

Artur S. d'Avila Garcez
Department of Computing
City University London
EC1V 0HB, London, UK.
aag@soi.city.ac.uk

April 30, 2009

Abstract

This paper studies methodologically robust options for giving logical contents to nodes in abstract argumentation networks. It defines a variety of notions of attack in terms of the logical contents of the nodes in a network. General properties of logics are refined both in the object level and in the metalevel to suit the needs of the application. The network-based system improves upon some of the attempts in the literature to define attacks in terms of defeasible proofs, the so-called rule-based systems. We also provide a number of examples and consider a rigorous case study, which indicate that our system does not suffer from anomalies. We define consequence relations based on a notion of defeat, consider rationality postulates, and prove that one such consequence relation is consistent.

1 Introduction

An abstract argumentation network has the form (S, R), where S is a nonempty set of arguments and $R \subseteq S \times S$ is an attack relation. When $(x, y) \in R$, we say x attacks y.

The elements of S are atomic arguments and the model does not give any information on what structure they have and how they manage to attack each other.

The abstract theory is concerned with extracting information from the network in the form of a set of arguments which are winning (or 'in'), a set of arguments which are defeated (or are 'out') and the rest are undecided. There are several possibilities for such sets and they are systematically studied and classified. See Figure 1 for a typical situation. $x \to y$ in the figure represents $(x, y) \in R$.

A good way to see what is going on is to consider a Caminada labelling. This is a function λ on S distributing values $\lambda(x), x \in S$ in the set {in, out, ?} satisfying the following conditions.

1. If x is not attacked by any y then $\lambda(x) = 1$

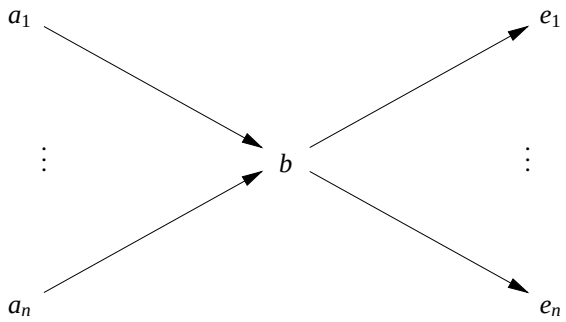


Figure 1:

2. If $(y, x) \in R$ and $\lambda(y) = 1$ then $\lambda(x) = 0$
3. If all y which attack x have $\lambda(y) = 0$ then $\lambda(x) = 1$.
4. If one y which attack x has $\lambda(y) =?$ and all other y have $\lambda(y) \in \{0, ?\}$ then $\lambda(x) =?$.

Such λ exist whenever S is finite and for any such λ, the set $S^+_\lambda = \{x \mid \lambda(x) = 1\}$ is the set of winning arguments, $S^-_\lambda = \{x \mid \lambda(x) = 0\}$ is the set of defeated arguments and $S^?_\lambda = \{x \mid \lambda(x) =?\}$ is the set of undecided arguments.

The features of this abstract model are as follows:

1. Arguments are atomic, have no structure.
2. Attacks are stipulated by the relation R; we have no information on how and why they occur.
3. Arguments are either 'in' in which case all their attacks are active or are 'out' in which case all their attacks are inactive. There is no in between state (partially active, can do some attacks, etc.). Arguments can be undecided.
4. Attacks have a single strength, no degrees of strength or degree of transmission of attack along the arrow, etc.
5. There are no counter attacks, no defensive actions allowed or any other responses or counter measures.
6. The attacks from x are uniform on all y such that $(x, y) \in R$. There are no directional attacks or coordinated attacks.[1] In Figure 1, $a_1, \ldots, a_n$ attack b individually and not in coordination. For example, a_1 does not attack b with a view of stopping b from attacking e_1 but without regard to $e_1, \ldots, e_n$.

[1]There is some controversy on whether arguments accrue. While Pollock denies the existence of cumulative argumentation [15], Verheij defends that arguments can be combined either by subordination or by coordination, and may accrue in stages [16]. The debate is by no means over or out of date, see e.g. also [13]. Relatedly, in neural networks, the accrual of arguments by coordination appears to be a natural property of the network models [6]. The accrual of arguments can also be learned naturally by argumentation neural networks.

7. The view of the network is static. We have a graph here and a relation R on it. So Figure 1 is static. We use the words 'there is no progression in the network' to indicate this; the network is static. We seek a λ labelling on it and we may find several. In the case of Figure 1 there is only one such λ. $\lambda(a_i) = 1, \lambda(b) = 0, \lambda(e_j) = 1, i, j = 1, \ldots, n$.
   We advocate a dynamic view, like first a_i attack b and b then (if it is not out dead) tries to attack e_i. Or better still, at the same time each node launches an attack on whoever it can. So a_i attack b and b attacks e_i and the result is that a_i are alive (not being attacked) while b and e_j are all dead.

Points 4 and 7 above have been addressed in [2], and points 6 and 7 in [6], but points 1–3 and 5 remain untreated by us. It is our aim in this paper to give theoretical answers to these questions.

There are several authors who have already addressed some of these questions. See [3; 4]. We shall build upon their work, especially [4].

Obviously, to answer the above questions we must give contents to the nodes. We can do this in two ways. We can do this in the metalevel, by putting predicates and labels on the nodes and by writing axioms about them or we can do it in the object level, giving internal structure to the atomic arguments and/or saying what they are and defining the other concepts, e.g. the notion of attack in terms of the contents.

Example 1.1 (Metalevel connects to nodes) *Figure 2 is an example of a metalevel extension.*

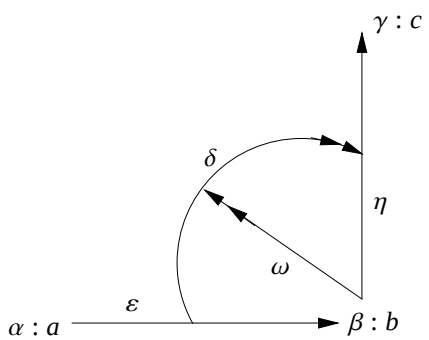


Figure 2:

The node a is labelled by α. It attacks the node b with transmission factor ε. This transmission factor *is an important feature of our approach. In fact, it will prove crucial in answering some of the questions. The idea stems from our research on neural-symbolic computation [8], where the weights of neural networks are always labelled by real numbers which are learnable (i.e. can be adapted through the use of a learning algorithm to account for a new situation).*

Node b is labelled by β. The attack arrow itself constitutes an attack on the attack *arrow from b to c. This attack is itself attacked by node b. Each attack has its own transmission factor. We denote attacks on arrows by double arrows. Allowing attacks*

on arrows is another new idea in argumentation, first proposed in the context of neural computation in [7]. It can be associated with the above-mentioned learning process, where an agent identifies the changes that are required in the system. This concept turns out to be quite general and yet useful in a computational setting. In the case of a recurrent network, for example, attacks on arrows can be used to control infinite loops, as discussed in [2] and exemplified through the use of learning algorithms in [6]. In other words, we see loops as a trigger for learning.

Formally, we have a set S of nodes, here

$$S = \{a, b, c\}.$$

The relation R is more complex. It has the usual arrows $\{(a, b), (b, c)\} \subseteq R$ and also the double arrows, namely, $\{((a, b), (b, c)), (b, ((a, b), (b, c)))\} \subseteq R$. We have a labelling function $\mathbf{l}$, giving values

$$\begin{array}{l} \mathbf{l}(a) = \alpha, \mathbf{l}(b) = \beta, \mathbf{l}(c) = \gamma, \\ \mathbf{l}((a, b)) = \varepsilon, \mathbf{l}((b, c)) = \eta, \\ \mathbf{l}(((a, b), (b, c))) = \delta \\ \mathbf{l}((a, ((a, b), (b, c)))) = \omega. \end{array}$$

We can generalise the Caminada labelling as a function from $S \cup R$ to some values which satisfy some conditions involving the labels. We can write axioms about the labels in some logical language and these axioms will give more meaning to the argumentation network. See [2] for some details along these lines. The appropriate language and logic to do this is Labelled Deductive Systems (LDS) [9].

We shall not pursue the metalevel extensions approach in this paper except for one well known construction which will prove useful to us later.

Example 1.2 (The logic program associated with an ordinary abstract network) *Let $N = (S, R)$ and consider S as a set of literals. Let $\Rightarrow$ be the logic programming arrow and let $\wedge, \neg$ be conjunction and negation as failure. Consider the logic program $P(N)$ containing the following clauses C_x, $x \in S$*

$$C_x : \bigwedge_{i=1}^{m} \neg y_i \Rightarrow x$$

where $y_1, \ldots, y_m$ are all the nodes in S which attack x (i.e. $(\bigwedge_{(y,c)\in R} y) \Rightarrow x$).

If no node attacks x then $C_x = x$.

C_x simply says in logic programming language that x is in *if all y which attack it are* out *(i.e. $\neg y_i$).*

In [14], a neural network is used as a computational model for conditional logic, in which *attacks on arrows* are allowed. More precisely, these are graphs where arcs are allowed to connect not only nodes, but nodes to arcs, denoting an exception that can change a default assumption. For example, suppose that node a is connected to node b, indicating that *a normally implies b*. A node c can be connected to the connection from

a to b, indicating that c is an exception to the rule. In other words, if c is activated, it blocks the activation of b, regardless of the activation of a. In logic programming terms, we would have $a \wedge \neg c \Rightarrow b$. Leitgeb's networks can be reduced to networks containing no arcs connected to arcs; these are the *CILP* networks used in [5] to compute and learn logic programming. Here, the networks are more general. There are three cases to consider:

1. The fact that a node a attacks a node b can attack a node c, $(a \to b) \to c$;

2. A node a can attack the attack of a node b on a node c, $a \to (b \to c)$; and

3. The fact that node a attacks node b attacks the attack from node c to node d (but not any other attack on d), $(a \to b) \to (c \to d)$.

Here, there are cases that cannot be reduced or flattened. The most general network set-up allowing for connections to connections is the fibring set-up of [7], where it is proved that fibred networks are strictly more expressive than their flattened counterpart, *CILP* networks. In [7], nodes in one network (or part of a network) are allowed to change dynamically the weights (or transmission factors) of connections in another network. This can be seen as an integration of learning (the progressive change of weights) into the reasoning system (the network computation). It provides a rich connectionist model towards a unifying theory of logic and network reasoning.

We are now ready for our second approach, namely giving logical content to nodes.

Assume we are using a certain logic **L**. **L** can be monotonic, nonmonotonic, algorithmic, etc. At this stage anything will do. This logic has the notion of formulas A of the logic, theories Δ of the logic and the notion of $\Delta \vdash A$, and possibly also the notion of Δ is not consistent.

The simplest approach is to assume the nodes $x \in S$ are theories Δ_x supporting logically a formula A_x (i.e. $\Delta_x \vdash A_x$ in the logic). The exact nature of the nodes will determine our options for defining attacks of one node on another.

We list the important parameters.

1. The nature of the logic at node x and how it is presented. The logic can be classical logic, intuitionistic logic, substructural logic, nonmonotonic logic, etc. It can be presented proof theoretically, or semantically or as a consequence relation, or just as an algorithm.

2. What is Δ_x? A set of wffs? A proof? A network (e.g. a Bayesian network) with algorithms to extract information from it? etc.

3. The nature of the support Δ_x gives A_x. We can have $\Delta_x \vdash A_x$, or we can have that A_x is extracted from Δ_x by some algorithm $\mathcal{A}_x$ (e.g. abduction algorithms, etc.).

4. How does the node x attack other nodes? Does it have a stock of attack formulas $\{\alpha_1, \alpha_2, \ldots\}$ that it uses? Does it use A_x? etc.

5. What does the node x do when it is attacked? How does it respond? Does it counter attack? Does it transform itself? Does it die (become inconsistent)?

6. To define the notion of an attack one must give precise formal definitions of all the parameters involved.

We give several examples of network and attack options.

Example 1.3 (Networks based on monotonic logic) *Let* **L** *be any monotonic logic, with a notion of inconsistency. Let the nodes have the form* $x = (\Delta_x, A_x)$ *where* Δ_x *is a set of formulas such that* $\Delta_x \vdash A_x$ *and* Δ_x *is a minimal such set (i.e. no* $\Theta \subsetneq \Delta_x$ *can prove* A_x*).*

Δ_x *attacks* Δ_y *by forcing itself onto* Δ_y *(i.e. forming* $\Delta_x \cup \Delta_y$*). If the result is inconsistent then a revision process starts working and a maximal* $\Theta_y \subseteq \Delta_y$ *is chosen such that* $\Delta_x \cup \Theta_y$ *is consistent. The result of the attack on* y *is* Θ_y*. Of course if* $\Delta_x \cup \Delta_y$ *is consistent then the attack fails, as* $\Theta_y = \Delta_y \vdash A_y$*, otherwise* $\Theta_y \nvdash A_y$ *and the attack succeeds. However, the node* y *transforms itself into a logically weaker node.*

Note that unless Θ_y *is empty, the new transformed* Θ_y *is still capable of attacking. To give a specific example, consider the two nodes:*

$$x = (\neg A, \neg A) \text{ and } y = ((A, A \to B), B)$$

x *attacks* y *and the result of the attack is a new* $\Delta'_y = \{A \to B\}$*.* Δ'_y *can still attack its targets though with less force.*

Consider the following sequence:

$$z = (\neg E, \neg E), x' = ((E, \neg A), \neg A \wedge E), y = ((A, A \to B), B)$$

z *attacks* x'*,* x' *as a result of the attack regroups itself into* x *and proceeds to attack* y*. Note that we view progression from left to right along* R*.*

Consider now the following Figure 3, as a third example:

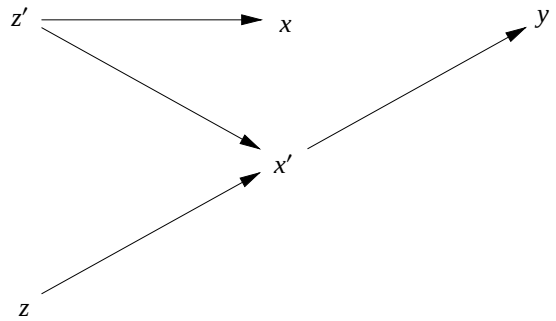


Figure 3:

where $z' = (A, A)$ *and* z, x', x *and* y *are as before. Because of the attack of* z'*,* x' *cannot regroup itself into* x *because* x *is also being attacked.*

Consider now a fourth example, Figure 4. Here neither x_1 *nor* x_2 *can cripple* y *but a joint attack can.*

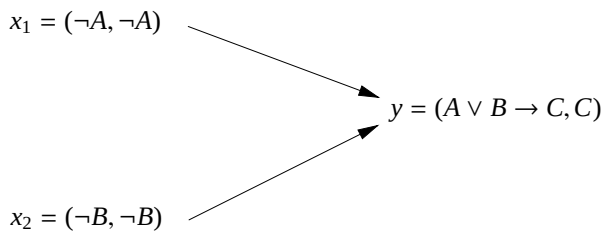


Figure 4:

Remark 1.4 (Summary of options for the monotonic example)

1. *Attacks are done by hurling oneself at the target. This can be refined further by allowing sending different formulas at different targets.*
2. *Attacks can be combined.*
3. *The target may be crippled but can still 'regroup' and attack some of its own targets.*
4. *The nature of any attack is based on inconsistency and revision.*
5. *We can sequence the attacks as a progression along the relation R.*
6. *Attacks are not symmetrical since we use revision. So if A attacks $\sim A$, AGM revision [1], for example, will give preference to A. So for $\sim A$ to attack A it has to do so explicitly, and the winner is determined by the progression of the attack sequence.*

Example 1.5 (Networks based on nonmonotonic logic) *This example allows for nodes of the form $x = (\Delta_x, A_x)$ where the underlying logic is a nonmonotonic consequence $\mid\!\sim$. In nonmonotonic logic we know that we may have $\Delta_y \mid\!\sim A_y$ but $\Delta_y + B \not\mid\!\sim A_y$.*[2]

So if node $x = (\Delta_x, A_x)$ attacks node y, it simply adds Δ_x to Δ_y and we get $y' = \Delta_x \cup \Delta_u \mid\!\sim ?A_y$.

Here the attack is based on providing more information and not on inconsistency and revision.

To show the difference, let Δ_y be:

1. *Bird* $(a) \mapsto$ *Fly* (a)
2. *Penguin* $(a) \wedge$ *Bird* $(a) \mapsto \sim$ *Fly* (a)

[2] A nonmonotonic consequence on the wffs of the logic satisfies three minimal properties

1. Reflexivity: $\Delta \mid\!\sim A$ if $A \in \Delta$.
2. Restricted monotonicity: $\Delta \mid\!\sim A$ and $\Delta \mid\!\sim B$ imply $\Delta, A \mid\!\sim B$.
3. Cut Rule: $\Delta, A \mid\!\sim B$ and $\Delta \mid\!\sim A$ imply $\Delta \mid\!\sim B$.

Note that $\mid\!\sim$ can be presented in many ways, semantically, proof theoretically or algorithmically.

3. *Bird* (a)

where $\mapsto$ *is defeasible implication.*

Let A_y *be Fly* (a).

Let Δ_x *and* A_x *be Penguin* (a). *x can attack y by sending it the extra information that Penguin* (a). *Another attack from another point* x' *to y can be by sending* $\sim$ *Bird*(a) *to y, i.e.* $\Delta_{x'} = A_{x'} = \sim$ *Bird* (a).

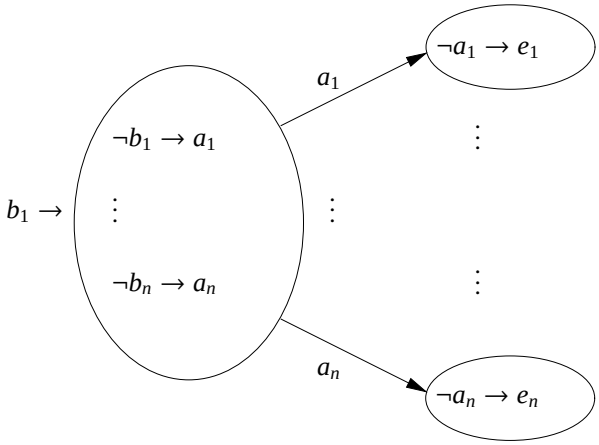


Figure 5:

Example 1.6 (Prolog programs) *The theories here are Prolog programs and the arguments are the literals they prove.*

An attack is executed by sending a literal from the attacking theory to the target theory. See Figure 5.

Example 1.7 (Counter-attack) *The Dung framework does not allow for counter-attacks being effective only when attacked but not before. The model is static. The attacks do not 'progress' along the network like a flow going through the nodes activating them as it goes along. However, if we perceive such progression, we can define the concept of counter-attack. This is the same progression that may resolve syntactic loops in [6].*

Consider Figure 6. $\Delta_1 \vdash a$ *and can attack* Δ_2 *by passing a along the attacking arrow. The* $?d$ *is a counter-attack. As long as* Δ_2 *is not attacked by* Δ_1, *d is not provable and so cannot be sent to* Δ_1. *Once* Δ_2 *is attacked then d becomes provable and can counter-attack* Δ_1 *and render a unprovable.*

Example 1.8 (Directional attacks) *The following is a more enriched logical model where more options are naturally available. We can let a node a be a nonmonotonic theory* Δ_a *such that* $\Delta_a \mid\!\sim a$. *We can understand an attack of a nonmonotonic node a, say, on node* e_1 *as the transmission of an item of data, say* α_1 *(such that* $\Delta_a \mid\!\sim \alpha_1$*) to* Δ_{e_1} *with the effect that* $\Delta_{e_1} + \alpha_1 \not\mid\!\sim e_1$. *Since* Δ_{e_1} *is nonmonotonic, the insertion of* α_1 *into it may change what it can prove. See Figure 7.*

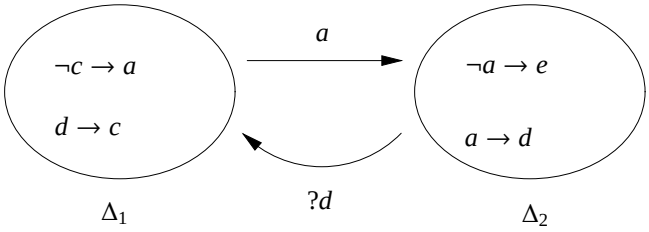


Figure 6:

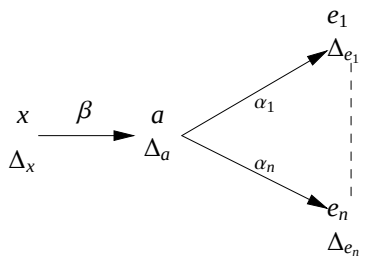


Figure 7:

We have $\Delta_x \mid\!\sim \beta, \Delta_x \mid\!\sim x, \Delta_a \mid\!\sim a, \Delta_a \mid\!\sim \alpha_i, i = 1, \ldots, n$.
We may have

$$\Delta_a + \beta \not\mid\!\sim \alpha_1$$

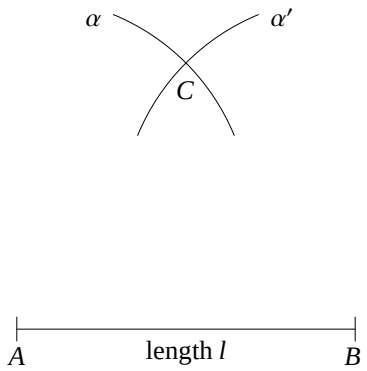
and therefore the attack on e_1 *fails, but we may still have that* $\Delta_a + \beta \mid\!\sim \alpha_n$*, hence the attack on* e_n *still succeeds. The attack by* β *is not a specific attack on the arrow from a to* e_1*. It tansforms a to something else which does not attack* e_1*. So Figure 7 is not a good representation of it. It shows the result but not the meaning.*

By the way, to attack the attack from x to a in Figure 7, we might add a formula β' to β, and so the attack changes from β to (β and β').

Example 1.9 (Abduction) *Another example can be abduction.*

The node y contains an argument of the following form. It says, we know of Δ_y *and the fact that a formula* E_y *should be provable, but* Δ_y *cannot prove* E_y*. So we abduce* A_y *as the most reasonable additional hypothesis. So the node y is* (Δ_y, A_y)*, where* $A_y = \text{Abduce}(\Delta_y, E_y)$*. x can attack by sending additional information* Δ_x*. It may be that* $\Delta_y \cup \Delta_x \not\vdash E_y$*, but* $\text{Abduce}(\Delta_y \cup A_x, E_y)$ *is some* A'_y *and not* A_y*.*

An example that we like is from Euclid. Euclid proved that if we have a segment of length l we can construct a triangle ABC whose sides are all equal to length l. The construction is as in the diagram:



We construct the two arcs α and α' of radius l around A and B and they intersect at point C.

The gap in the proof is that the two arcs may slip through gaps in each other. In other words the point C may be a hole in the plane. The principle of minimal hypothesis for abduction would add the least hypothesis needed namely that all rational points in the field with $\sqrt{2}$ are allowed but not more. So the lines can still have gaps in them. This argument can be attacked by the additional information that the Greeks thought in terms of continuous lines and not in terms of the field generated by the rationals and $\sqrt{2}$. So we must abduce the hypothesis that lines have no gaps. Computationally this may, of course, be problematic still. Discrete algorithms cannot deal with continuous lines having an infinite number of points; some approximation will be necessary.

Example 1.10 (Replacement networks) *Let $\mid\!\sim$ be a consequence relation. Consider a network $N = (S, R)$, where the set S contains atoms of the language of $\mid\!\sim$ and the nodes x have the theories (Δ_x, A_x) associated with them, where $\Delta_x = \{x\}$ and $A_x = x$.*

The network (S, R) can now be viewed in two ways. One as an abstract network and one as a logical network with (Δ_x, A_x).

To have the two points of view completely identical we must assume about $\mid\!\sim$ that the following holds:

(*) whenever $y_1, \ldots, y_k$ are all the nodes that attack x (i.e. $y_i R x$ holds) then we have $\{y_i, x\} \not\mid\!\sim x$, for each $i, i = 1, \ldots, k$.

When we have (*) the logical attack coincides with the abstract network attack. By the properties of consequence relation, we also have $y_i \not\mid\!\sim x$. Note that we do not know much about $\mid\!\sim$ beyond property (*) and so any nonmonotonic consequence relation satisfying (*) will do the job of being equivalent to the abstract network. So let us take a Prolog consequence $\Vdash$ for a language with atoms, $\wedge, \Rightarrow$ and $\neg$ (negation as failure). Let

$$\Delta_x = (\bigwedge \neg y_i) \Rightarrow x$$
$$A_x = x$$

This $\Vdash$ satisfies condition (*) and so can represent or replace $\mid\!\sim$ on the networks. Compare with Example 1.2.

Remark 1.11 *Example 1.10 leaves us with several general questions*

1. *Given a nonmonotonic $\mid\!\sim$ under what conditions can it be represented by a Prolog program?*
2. *What can we do with extensions of Prolog, say N-Prolog [10], etc. How much more can we get?*
3. *Given the above network, what do we get if we describe it in the meta-level as we did in Example 1.2*
4. *Given a reasonable $\mid\!\sim$, can we cook up a reasonable extension of Prolog to match it? Can we be systematic about it and have the same construction for any reasonable $\mid\!\sim$?*

2 Methodological considerations

In order to present a methodologically robust view of logical modes of attack in argumentation networks, as intutiively described in the last section, we need to clarify some concepts. There is logical tension between two possibly incompatible themes.

Theme 1

Start with a general logical consequence $\mid\!\sim$, use databases of this logic as nodes in a network and define the notion of attack and then emerge with one or more admissible extensions.

Questions

These extensions are sets of nodes (the 'winning' nodes or the network 'output' nodes). They contain logic in them, being themselves theories of our background logic. What are we going to expect from them? Consistency? Are we going to define a new logic from the process?

Theme 2

We start with some notion of proof (argument). We can prove opposing formulas or databases of some language **L**. We create a network of all the proofs we are interested in and define the notion of one proof (argument) attacking another. We emerge with several admissible or winning sets of proofs.

Questions

What are we to do with these proofs? Do we define a logical consequence relation using them? For example, let Δ be a set of formulas and rules. Let S be all possible proofs we can build up using Δ. Note that these proofs can prove opposing results, e.g. q and $\sim q$, etc. So we do not yet have a consequence relation for getting results out of

Δ. Let R be a notion of attack we define on S. Let E be a winning extension chosen in some agreed manner. Then we define a new consequence by declaring $\Delta \mid\!\sim E$.

What connection do we require between this new consequence $\mid\!\sim$ and some other possibly reasonable consequence relation we can define directly using proofs (without the intermediary of networks)? We need rationality postulates on the notion of defeat.

To make the above questions precise and gain some intuitions towards their solutions we need to examine some examples in rigorous detail. We begin with some puzzles critically examined in [4].

Example 2.1 *This is example 4 in [4, p. 292]. The language allows for atoms, negation $\sim$, strict rules (implication) $\rightarrow$ and defeasible rules (implication) $\Rightarrow$. The theory Δ contains*

1. *wr (strict fact)*
   Reading: John wears something that looks like a wedding ring.

2. *go (strict proof)*
   Reading: John often goes out until late with his friends.

3. $wr \Rightarrow m$
   m reads: John is married

4. $go \Rightarrow b$
   b reads: John is a bachelor

5. $m \rightarrow hw$
   hw reads: John has a wife

6. $b \rightarrow\sim hw$

If modus ponens (detachment) is the only rule we can use, we can construct the following arguments from Δ :

$$\begin{array}{ll} A_1: & wr \\ A_2: & go \\ A_3: & wr, wr \Rightarrow m \\ A_4: & go, go \Rightarrow b \\ A_5: & wr, wr \Rightarrow m, m \rightarrow hw \\ A_6: & go, go \Rightarrow b, b \rightarrow\sim hw. \end{array}$$

The following is implicit in the Caminada and Amgoud understanding of the situation.

I1 $\Delta = \{1, 2, 3, 4, 5, 6\}$

I2 *The argument network is* all *possible arguments that can be constructed from elements of Δ.*

I3 *An argument is a sequence (chain) or elements from Δ that respect modus ponens (detachment).*

I4 *Given $x \rightarrow y$, we take it literally and do not say that we also have $\sim y \rightarrow \sim x$. If we want the latter we need to include it explicitly. This assumption is clear since later Caminada and Amgoud do include such additional rules explicitly as part of their proposed solution to some anomalies.*

I5 *One argument attacks another if the last head of the last implication is the negation of the last head of the last implication of the other.*

Caminada and Amgoud point out an anamoly in this example. They point out that $A_1, \ldots, A_4$ do not have any defeaters. So they win. What $\{A_1, \ldots, A_4\}$ prove (their 'output' as they define it), is the set $\{wr, go, m, b\}$. Thus if the output is supposed to mean what is justified then both m and b are to be considered justified. Yet, and here is the anomaly, the strict rules closure of the output is inconsistent since it contains $\{hw, \sim hw\}$.

We now discuss this example.

First let us try to sort out some confusion. Are we working in Theme 1, where there is a background logic or in Theme 2 where we want to use an argumentation framework to define a logic?

If there is a background logic then does it inlcude closure under strict rules? If yes, then A_3 and A_4 already attack each other. If no, then don't worry about the inconsistency of the output. We simply have defined an inconsistent theory using the tool of argumentation networks.

Caminada and Amgoud are aware that if we allow closure under strict rules at every stage then the anomaly is resolved. They attribute this solution to Prakken and Sartor [17]. They offer another example, which has anomaly, example 6, page 293, and where this trick does not work. We shall address this example later. Let us first consider Caminada and Amgoud's own solution to Example 4. They add two more contraposition rules to the database.

7. $hw \rightarrow \sim b$

8. $\sim hw \rightarrow \sim m$

With two more rules in the database, two more arguments can be constructed from the database:

A7 $wr, wr \Rightarrow m, m \rightarrow hw, hw \rightarrow \sim b$

A8 $go, go \rightarrow b, b \Rightarrow \sim hw, \sim hw \rightarrow \sim m.$

Now that our stock of arguments has A7 and A8, we have that A8 defeats A3 and A7 defeats A4. The set of winning arguments changes and the only justified arguments are $\{wr, go\}$, without the anomalies $\{b, m\}$.

We do not consider this as a solution to the anomaly. Caminada and Amgoud changed the problem (i.e. took a different, bigger database) and changed the underlying

logic. Not always do we have that if $x \rightarrow y$ is a rule so is $\sim y \rightarrow \sim x$. We need to give a rigorous definition of the defeasible logic we are using, and then examine the problem of anomalies. We shall do this in Section 3. See Example 3.9 and Remark 3.12. The anomalies arise because the Dung framework does not allow for joint attacks. By the way, Caminada and Amgoud have done an excellent analysis of the anomalies. We are simply continuing their initial work.

Let us now address Example 6 of [4].

Example 2.2 *The database has the following facts and rules*

1. *a, strict fact*
2. *d, strict fact*
3. *g, strict fact*
4. $b \wedge c \wedge e \wedge f \rightarrow \sim g$
5. $a \Rightarrow b$
6. $b \Rightarrow c$
7. $d \Rightarrow e$
8. $e \Rightarrow f$.

Caminada and Amgoud consider the following arguments

A: $a, a \Rightarrow b$

B: $d, d \Rightarrow e$

C: $a, a \Rightarrow b, b \Rightarrow c$

D: $d, d \Rightarrow e, e \Rightarrow f$.

We also have the arguments

F1: a

F2: d

F3: g

The notion of one argument defeating another is the same as before, i.e. we need the two arguments to end their chains with opposite heads. Thus A, B, C, D do not have any defeaters. The justified literals are $\{b, c, e, f\}$ as well as the facts $\{a, d, g\}$.

Thus we get an anomaly: the closure of the winning facts under strict rules is not consistent.

We again ask the question, what exactly is the underlying logic? We need a formal definition to assess the situation.

Is the following argument G also acceptable?

G: $A, C, B, D, 4$

In other words, G is

$$a, a \Rightarrow b; a, a \Rightarrow b, b \Rightarrow c; d, d \Rightarrow e; d, d \Rightarrow, e \Rightarrow f, b \wedge c \wedge e \wedge f \rightarrow \sim g$$

We first use A, C, B, D *to prove the antecedent of 4 and then get* $\sim g$.

If this argument is acceptable, then it must be included in the network, as the rules of the game is to include in the network all arguments which can be constructed from Δ*, then G and* $F3$ *attack each other and so the winning set is only* $\{b, c, e, f, a\}$ *and we have no inconsistency.*

If argument G is not acceptable because we cannot do modus ponens with more than one assumption, then the winning set is indeed $\{b, c, e, f, a, g\}$ *but then we cannot get inconsistency because we cannot use modus ponens with* $b \wedge c \wedge e \wedge f \rightarrow \sim g$.

So again we ask: we need a rigorous definition of the logic!

Depending on how the logic works, we may be able to deduce, for example, from A the rule $c \wedge e \wedge f \Rightarrow \sim g$ *(since b defeasibly follows from a) and similarly from B we deduce* $b \wedge c \wedge f \Rightarrow \sim g$ *and from C we get* $e \wedge f \Rightarrow \sim g$ *and from D we get* $b \wedge c \Rightarrow \sim g$.

If we are allowed to have that then we have that C and D defeat each other, and again we have no anomaly. So it all depends on the logic.

It would be better to compute these arguments using the network itself, as we have done in [6] for a simpler argumentation framework (where arguments are atomic rather than proofs). We are working on this for the general case. We believe that network fibring has the answer [11; 7].

Let us now define one such a logic. We shall indicate what options we have.

Definition 2.3 *Let Q be a set of atoms. Let* $\wedge$ *be conjunction,* $\sim$ *a form of negation and* $\rightarrow$ *stand for strict (monotonic) implication and* $\Rightarrow$ *for defeasible implication.*

2. *A rule has the form*
   $\pm a_1 \wedge \ldots \wedge \pm a_n \rightarrow \pm b$ *(strict rule)*
   $\pm a_1 \wedge \ldots \wedge \pm a_n \Rightarrow \pm b$ *(defeasible rule)*
   where a_i, b *are atoms,* $+a$ *means a and* $-a$ *means* $\sim a$.

3. *A fact has the form* $\pm a$ *(we consider strict facts only; an alternative would be to consider beliefs, and yet another to consider degrees of belief).*

4. *A database* Δ *is a set of rules (strict or defeasible) and facts.*

Definition 2.4 *Let* Δ *be a database. We define the notion of the sequence* π *of formulas (actually a tree of formulas written as a sequence) is an argument for the literal* $\pm a$ *of length n and defeasible degree m, and specificity* σ.

1. π *is an argument of* $\pm a$ *from* Δ *of length 1 and degree 0 iff* $\pm a \in \Delta$ *and* $\pi = (\pm a)$. *Let* $\sigma = \{\pm a\}$.

2. *Assume $\pi_1, \ldots, \pi_k$ are all proofs of $\pm a$ from Δ of lengths n_i and degree m_i and specificity sets σ_i resp. for $i = 1, \ldots, k$. Assume $\bigwedge \pm a_i \rightarrow \pm b$ is a strict rule. Then $(\pi_1, \pi_2, \ldots, \pi_n, \bigwedge \pm a_i \rightarrow \pm b)$ is an argument for $\pm b$ of length $1 + \sum_i n_i$ and degree $\mathbf{f}_{\rightarrow}(m_1, \ldots, m_k)$, where $\mathbf{f}_{\rightarrow}$ is some agreed function representing the degree of 'defeasiblility' in the argument.*

   Options for $\mathbf{f}_{\rightarrow}$ are

   Option max
   $\mathbf{f}_{\rightarrow} = max(m_i)$

   Option sum[3]
   $\mathbf{f}_{\rightarrow} = \sum m_i$

   Let $\sigma = \bigcup_{i=1}^{k} \sigma_i$.

3. *Assume π_i are arguments of $\pm a_i$. Let $\bigwedge \pm a_i \Rightarrow \pm b$ be a defeasible rule. Then $\pi_1, \ldots, \pi_k, \bigwedge \pm a_i \rightarrow \pm b$ is an argument of $\pm b$. The length of the argument is $1 + \sum n_i$ and the degree of the argument is $\mathbf{f}_{\Rightarrow} = 1 + \mathbf{f}_{\rightarrow}(m_1, \ldots, m_k)$, and $\sigma = \bigcup \sigma_i$.*

Remark 2.5

1. *The strict rules are not necessarily classical logic. So for example from $x \rightarrow \sim y$ and y we cannot deduce $\sim x$.*

2. *The definition of an argument watched for the complexity m measuring how many defeasible rules are used in the argument and the specificity σ recording the set of literals (i.e. the factual information) used in the argument. This measure is used later to define when one argument defeats another. We know from defeasible logic that the specificity of a rule is also important. So $a \wedge b \Rightarrow c$ is more specific that $a \Rightarrow c$. The set σ is a rough measure of specificity. One can be more fine tuned. We can define a more complex measure say μ which reflects a finer balance between the number of defeasible rules used and their specificity.*

Definition 2.6 *Let Δ be a database. An argument π is said to be based on Δ if all its elements are in Δ. We now define the notion of $\Delta \vdash\sim \pm a$, a atomic, using Theme 1 point of view.*

We wish to do this in steps:

Step 1 $\Delta \vdash_1 \pm a$ *iff* $\pm a \in \Delta$

Step $m+1$ $\Delta \vdash_{m+1} \pm a$ *iff there is a rule in Δ of the form $\bigwedge \pm a_i \Rightarrow \pm a$ such that $\Delta \vdash_{m_i} \pm a_i$, with $\sum m_i = m$, and for no rule in Δ of the form $\bigwedge \pm a'_i \Rightarrow \mp a$ (note $\mp a = \sim \pm a$) do we have $\Delta \vdash_{m'_j} \pm b_j$ with $\sum m'_i < m$, or if $\sum m'_i = m$ then we do not have that $\bigcup \sigma_i \subsetneq \bigcup \sigma'_i$. (In words, $\pm a$ is proved using defeasible rule complexity m and specificity set σ and there is neither a less complex*

[3]One could think of this as: the more involved the proof, the weaker the argument. For example, the more steps there are in the proof, the larger $\sum m_i$. Notice that a fact is strongest.

argument for $\mp a$ nor an argument for $\mp a$ with the same complexity but more specific, i.e. $\sigma \subsetneq \sigma'$.)

We also agree that if $\Delta \mid\!\sim_m \pm a_i$ and $\bigwedge \pm a_i \rightarrow \pm a \in \Delta$ then $\Delta \mid\!\sim_m \pm a$. We say $\Delta \mid\!\sim \pm a$ if $\Delta \mid\!\sim_m \pm a$ for some m.

Remark 2.7 *The previous definition is one possibility of many. The important point to note is that any definition of $\mid\!\sim$ must say inside the induction step how one argument defeats another.*

Let us give some examples.

Example 2.8

1. *Let Δ be $\{d, a, a \Rightarrow b, d \Rightarrow \sim c, d \wedge b \Rightarrow c\}$.*

   We have that $\Delta \not\mid\!\sim_2 c$ because the argument $a, a \Rightarrow b, d, d \wedge b \Rightarrow c$ is defeated by the argument $d, d \Rightarrow \sim c$, and thus $\Delta \mid\!\sim \sim c$.

   Some defeasible systems will say the argument for c defeats the argument for $\sim c$ because it is more specific. Our system says the argument for $\sim c$ defeats the argument for c because it uses fewer defeasible rules.

2. *Our definition does say, for example, that for the database $\Delta' = \{a, d, a \Rightarrow c, a \wedge d \Rightarrow \sim c\}$ we have that $\sim c$ can be proved because it relies on more specific information. See remark 2.5.*

3. *We could give a definition which measures not only how many defeasible rules are used but also gives them weights according to how specific they are. Our aim here is not to develop the theory of defeasible systems and their options and merits but simply to show how one defines the notion of defeasible consequence relation and to make a single most important point:*

   > *To define the notion of consequence relation for a defeasible system we must already have a clear notion of argument defeat.*

Definition 2.9 *We now give a second definition of a consequence relation:*

1. *Let A, B be two arguments. Define the notion of A defeats B in some manner. Denote it by $A\mathcal{D}B$.*

2. *Let Δ be a theory, being a set of rules and literals. Let N be the set of all arguments based on Δ. Consider the network $(N, \mathcal{D})$ where $\mathcal{D}$ is the relation from (1) above. Let $\mathcal{A}$ be an algorithm for choosing a winning justified set of atoms from the net, e.g. let us $\mathcal{A}$ take the unique grounded extension which always exists. Then define $\Delta \mid\!\sim_{\mathcal{D},\mathcal{A}} \pm a$ iff $\pm a$ is justified by the above process $\mathcal{A}$ in the $(N, \mathcal{D})$ network.*

We are now ready for some methodological comments.

Rationality postulates for defeat

We need rationality postulates on the notion $\mathcal{D}$ of one argument defeating another where the arguments are defined in the context of facts, strict rules and defeasible rules. Caminada and Amgoud give rationality postulates on the admissible sets derived from $\mathcal{D}$ but this is insufficient. $\mathcal{D}$ must be such that it ensures we get a proper consequence relation $\mid\!\sim_{\mathcal{D}}$ out of it, satisfying reflexivity, restricted monotonicity and cut.

Representation problem

1. Given a consequence relation $\mid\!\sim$ for defeasible logic (i.e. $\mid\!\sim$ contains defeasible and strict rules), can we extract from $\mid\!\sim$ a defeat notion $\mathcal{D} = \mathcal{D}_{\mid\!\sim}$ for arguments, and a network algorithm $\mathcal{A}$ such that the notion $\mid\!\sim_{\mathcal{D},\mathcal{A}}$ is a subset of $\mid\!\sim$?

2. Given any consequence relation defined by any means (e.g. defined semantically), can we guess/invent a notion of argument and a notion of defeat $\mathcal{D}$ such that the associated $\mid\!\sim_{\mathcal{D},\mathcal{A}}$ is a subset of $\mid\!\sim$?

3. If we don't have such a representation theorem in the case of (1) above, using a natural $\mathcal{D}_{\mid\!\sim}$, then we perceive this as an anomaly.

Any solution to the anomalies raised in [4] must respect the above methodologial observations. It must not be an *ad hoc* solution.

3 A rigorous case study — 1

This section shows in a rigorous way how Theme 2 works. We define a nonmonotonic consequence relation using networks on arguments built up using rules.

Two comments

1. The strict rules need not be classical logic.

2. We use labelling to keep control of the proof process and possibly add strength to rules. However, the labels will not be used at first in our definitions and examples.

   Some strict logics require the labels in their formulation (e.g. resource logics) as well.

Definition 3.1

1. *Let our language contain atomic statements $Q = \{p, q, r, \ldots\}$, the connective $\sim$ for negation, $\wedge$ for conjunction, $\rightarrow$ for strict rules and $\Rightarrow$ for defeasible rules.*

2. *A literal x is either an atom q or $\sim q$. We write $-x$ to mean $\sim q$ if $x = q$ and q if $x =\sim q$.*

   A rule has the form $(x_1, \ldots, x_n) \rightarrow x$ (strict rule) or $(x_1, \ldots, x_n) \Rightarrow x$ (defeasible rule) where x_i, x are literals. We are writing $(x_1, \ldots, x_n) \rightarrow x$ instead of $\bigwedge x_i \rightarrow x$

to allow us to regard the antecedent of a rule as a sequence. This gives us a greater generality in interpreting the strict rules as not necessarily classical logic. We can also allow for $\emptyset \Rightarrow x$, where $\emptyset$ is the empty set.

3. *A rule of the form $(x_1, \ldots, x_n) \Rightarrow x$ is said to be more specific than a rule $(y_1, \ldots, y_m) \Rightarrow y$ iff $m < n$ and for some $i_1, \ldots, i_m \leq n$ we have $x_{i_j} = y_j$. Of course, any rule $(x_1, \ldots, x_n) \Rightarrow x$ is more specific than $\emptyset \Rightarrow y$. Note that we are not requiring $y =\sim x$.*

4. *A labelled database is a set of literals, strict rules and defeasible rules. We assume each element of the database has a unique label from a set of labels Λ. Λ is a new set of symbols, not connected with Q or anything else.*

   So we present the database as

   $$\Delta = \{\alpha_1 : A_1, \ldots, \alpha_k : A_k\}$$

   where α_i are different atomic labels from Λ and A_i are either literals or rules.

   The labels are just names at this stage, allowing us greater control of whatever we are going to do.

5. *Let Δ be a labelled database. We define by induction the strict closure of Δ denoted by Δ^S as follows:*

   (a) Let $\Delta_0^S = \Delta$.

   (b) Assume Δ_n^S has been defined. Let $\Delta_{n+1}^S = \Delta_n^S \cup \{\beta : x \mid$ for some $\alpha_i : x_i \in \Delta_n^S$, $\alpha : (x_1, \ldots, x_n) \rightarrow x \in \Delta$ and $\beta = (\alpha, \alpha_1, \ldots, \alpha_n)\}$.

   Let $\Delta^S = \bigcup_n \Delta_n^S$.

   Δ is consistent if for no literal x do we have $+x$ and $-x \in \Delta^S$.

6. *Note that we do not close under Boolean operations. The strict logic is not necessarily classical. We may have $\sim q \rightarrow r, \sim r \in \Delta$, this does not imply $q \in \Delta^S$.*

7. *Also note that only strict rules are used in the closure. So if Δ_0 is the set of defeasible rules in Δ, then $\Delta^S = \Delta_0 \cup (\Delta - \Delta_0)^S$.*

Definition 3.2 (Arguments) *We define the notion of an argument (or proof) π (based on a database Δ) its Δ-output $\theta_\Delta(\pi)$, its head $H(\pi)$, its literal base $L(\pi)$, and its family of subarguments $A(\pi)$.*

1. *Any literal $t : x \in \Delta$ is an argument of level 1. Its head is $t : x$. Its Δ-output is the set of all literals in the strict closure of $\{t : x\}$ and its head rule $H(\Delta)$ is $t : x$. Its literal base is $\{t : x\}$ and its subarguments are $\emptyset$.*

2. *Let $\pi_1, \ldots, \pi_n$ be arguments in Δ of level m_i, and let $\rho : (x_1, \ldots, x_n) \Rightarrow x$ be a defeasible rule in Δ. Assume $\alpha_i : x_i$ can be proved using strict rules from the union of the outputs $\theta_\Delta(\pi_i)$. Then $(\pi_1, \ldots, \pi_n, \rho : (x_1, \ldots, x_n) \Rightarrow x)$ is a new argument*

π. Its output is all the literals in the strict closure of $\{x\} \cup \bigcup_i \theta_\Delta(\pi_i), H(\pi) = \rho : (x_1, \ldots, x_n) \Rightarrow x$, $L(\pi) = \bigcup L(\pi_i)$, and $A(\pi) = \{\pi_1, \ldots, \pi_n\} \cup \bigcup_i A(\pi_i)$. The level of π is $1 + \max(m_i)$.

3. *An argument is consistent if its output is consistent.*

4. *Note that there is no redundancy in the structure of an argument. If a, b are literals then (a, b) is not an argument. If π_1, π_2 are arguments then (π_1, π_2) is not a argument.*

Definition 3.3 (Notion of defeat for arguments of levels 1 and 2) *Let π_1, π_2 be two consistent argument. We define the notion of π_1 defeats $\pi_2, \pi_1 \mathcal{D} \pi_2$, as follows:*

1. *A literal $t : x \in \Delta$ considered as an argument of level 1 defeats any argument π of any level 1 with $-x$ in its output. Note that if our arguments come from a consistent theory Δ, then no level 1 argument can defeat another level 1 argument. They are all consistent together as elements of Δ^S.*

2. *Let*
$$\pi_1 = (t_1 : x'_1, \ldots, t_n : x'_n, r : (x_1, \ldots, x_n) \Rightarrow x)$$
$$\pi_2 = (s_1 : y'_i, \ldots, s_m : y'_m, s : (y_1, \ldots, y_m) \Rightarrow y)$$
be two arguments of level 2, then π_1 defeats π_2 if $r : (x_1, \ldots, x_n) \Rightarrow x$ is more specific than $s : (y_1, \ldots, y_m) \Rightarrow y$, and $\theta_\Delta(\pi_2)$ and $\theta_\Delta(\pi_1)$ are inconsistent together.[4]

3. *In (2) above, we defined how an argument of level 2 can defeat another argument of level 2. (It cannot defeat any argument of level 1). Note that it can defeat an argument of any level m if it defeats any of its subarguments of level 2.*

4. *Note that two arguments of level 2 cannot defeat each other.*

5. *We shall give later the general definition of defeat for levels m, n.*

6. *An argument π_1 attacks an argument π_2 if*

   (a) *Their outputs are not consistent.*

   (b) *The head rule of π_1 is more specific than the head rule of π_2, or π_1 is of level 1.*

7. *π_1 may attack π_2 but not defeat it. However a level 2 argument always defeats other arguments it attacks.*

Example 3.4 *Let $\Delta = \{a, a \Rightarrow x, a \Rightarrow y, x \wedge y \rightarrow \sim a\}$. Δ is consistent because $\Delta^S = \{a\}$.*

The arguments
$$\pi_1 = (a, a \Rightarrow x)$$
$$\pi_2 = (a, a \Rightarrow y)$$

[4]Note that we do not require that $x =\sim y$, nor that $\{x, y\}$ is inconsistent. The requirement is that the outputs are inconsistent.

attack each other, but none can defeat the other because it has to be more specific.

Compare with Example 3.9.

Example 3.5 *This example does not use labels. We also write* $\bigwedge x_i \Rightarrow x$*, when we do not care about the order of* x_i*.*

1. *Consider the two arguments*

$$\pi_1 = (d, a, d \wedge a \Rightarrow c)$$
$$\pi_2 = (d, a, a \Rightarrow b, a \wedge b \wedge d \Rightarrow \sim c).$$

   π_1 *is of level 2 and* π_2 *is of level 3. In this section, our definition of defeat will say that* π_2 *defeats* π_1 *because the head of* π_2 *is more specific than the head of* π_1*. We are not giving advantage to* π_1 *on account of it being shorter (contrary to Definition 2.6).*

2. *Consider now* π_3

$$\pi_3 = (d, a, a \wedge d \Rightarrow \sim b, a \wedge d \Rightarrow c).$$

   Does π_2 *defeat* π_3*?*

   Its main head rule, $a \wedge b \wedge d \Rightarrow \sim c$ *is more specific but its subproof* $(d, a, a \Rightarrow b)$ *is defeated by the* π_3 *subproof* $(d, a, a \wedge d \Rightarrow \sim b)$*.*

So π_3 *defeats* π_2 *according to this section (as opposed to Definition 2.6).*

Example 3.6 (Cut rule) *Again we do not use labels, and we do not care about order in the antecedents of rules.*

Let Δ *be*

$$\Delta = \{b, d, d \wedge a \wedge b \Rightarrow c, d \Rightarrow c, a \wedge b \Rightarrow \sim c, c \Rightarrow a\}$$

We have

$$\Delta, a \mid\!\sim c$$

Because of the proof

$$\pi_1 : (b, d, a, d \wedge a \wedge b \Rightarrow c\},$$

$\pi_2 = (a, b, a \wedge b \Rightarrow \sim c)$ *is defeated by* π_1*.*

We also have

$$\Delta \mid\!\sim a$$

This is because of π_3*.*

$$\pi_3 = (d, d \Rightarrow c, c \Rightarrow a).$$

We ask do we have $\Delta \mid\!\sim c$*? We can substitute the proof of a into the proof of c, that is we substitute* π_3 *into* π_1*. We get* π_4*.*

$$\pi_4 = (b, d, d \Rightarrow c, c \Rightarrow a, d \wedge a \wedge b \Rightarrow c).$$

The question is, can we defeat π_4*? We can get a proof of* $\sim c$ *by substituting* π_3 *into* π_2*, to get* π_5

$$\pi_5 = (d, d \Rightarrow c, c \Rightarrow a, b, a \wedge b \Rightarrow \sim c).$$

Example 3.7 (Mutual defeat) *Let π_1 be $(a, b, a \wedge b \Rightarrow x)$. Let π_2 be $(a, b, c, a \Rightarrow \sim x, a \wedge b \wedge c \Rightarrow \sim x, \sim x \wedge \sim x \Rightarrow y)$. Then π_1 defeats a subargument of π_2, namely $(a, a \Rightarrow \sim x)$. A subargument of π_2, namely $(a, b, c, a \wedge b \wedge c \rightarrow \sim x)$ defeats π_1. You may ask why does π_2 prove $\sim x$ twice in two different ways? Well, maybe the strict rules of the logic are not classical and so two copies of $\sim x$ are needed (in linear logic $\sim x \rightarrow (\sim x \rightarrow y)$ is not the same as $\sim x \rightarrow y$), or maybe that is the way π_2 is; however, a proof is a proof.*

The output of π_1 is $\{a, b, x\}$ and the ouput of π_2 is $\{a, b, c, \sim x, y\}$. Each is consistent.

Definition 3.8 (Defeat for higher levels)

1. *We already defined how any argument of level 1 can defeat any argument of level $m \geq 2$. No argument of level m can defeat an argument of level 1 (this is because all arguments are based on a consistent Δ).*

2. *We defined how an argument of level 2 can defeat another argument of level 2.*

3. *An argument π_1 of level 3 can defeat an argument π_2 of level 2 if*

   (a) *one of its level 1 or level 2 subarguments defeats π_2*
   or

   (b) *its head is more specific than the head of π_2 of level 2, its output is inconsistent with the output of π_2, and π_2 does not defeat any of its level 2 subarguments.*

4. *Assume by induction that we know how an argument π_2 of level 2 can defeat or be defeated by an argument π_1 of level $k \leq m$. We show the same for level $m + 1$.*

   - *π_2 defeats π_1 if*

     (a) *π_2 defeats some subargument of level $k \leq m$ of π_1*
     or

     (b) *the head of π_2 is more specific than the head of π_1, its output is inconsistent with that of π_1, and no subargument of π_1 of level $\leq m$ defeats π_2.*

   - *The argument π_2 is defeated by π_1 if*

     (a) *Some subargument of π_1 of level $\leq m$ defeats π_2*
     or

     (b) *the head of π_1 is more specific than the head of π_2, its output is inconsistent with that of π_2, and π_2 does not defeat any subargument of level $\leq m$ of π_1.*

   We have thus defined how an argument of level 2 can defeat or be defeated by any argument of level m for any m.

5. *Assume by induction on k that we defined for level k and any m how any argument of level k can defeat or be defeated by any argument of level m for any m.*

   We define the same for level $k + 1$.

We define this by induction on m. We know from item (4) how π_{k+1} can defeat or be defeated by an argument of level 2. Assume we have defined how π_{k+1} can defeat or be defeated by any argument π'_n of level $n \leq m$. We define the same for level $n = m + 1$.

(a) π_{k+1} is defeated by an argument π'_{m+1} of level $m + 1$ if either π'_{m+1} defeats a subargument of π_{k+1} of level $\leq k$ or if the head of π'_{m+1} is more specific than the head of π_{k+1}, its output is not consistent with the output of π_{k+1} and no subargument of π'_{m+1} of level $\leq m$ is defeated by π_{k+1}.

(b) π_{k+1} defeats an argument π'_{m+1} if either it defeats a subargument of π'_{m+1} of level $\leq m$ or its head is more specific than the head of π'_{m+1}, its output is not consistent with the output of π'_{m+1} and no subargument of π_{k+1} of level $\leq k$ is defeated by π'_{m+1}.

6. *We thus completed the induction step of (5) and we have defined for any k and m how an argument of level k can defeat or be defeated by an argument of level m for any m and k.*

7. *We need one more clause: π_1 defeats π_2 if some subargument π_3 of π_1 defeats π_2 according to clause (1)–(5) above.*

Example 3.9 (Anomalies) *Consider the following database Δ.*

$$\begin{array}{rcl} \Delta & = & \{a, b, c, a \Rightarrow d, b \Rightarrow e, c \Rightarrow f, a \wedge b \wedge c \wedge d \wedge e \rightarrow\sim f\}. \\ \Delta^S & = & \{a, b, c\}. \end{array}$$

The arguments are, besides the literals a, b, c, the following:

$$\begin{array}{l} \pi_1 : a, a \Rightarrow d \\ \pi_2 : b, b \Rightarrow e \\ \pi_3 : c, c \Rightarrow f \end{array}$$

In our system, all the arguments form an admissible winning set and we get an anomaly since the output is inconsistent. We have no more arguments since we use in our definition only defeasible rules. If we allow in arguments for strict rules, or turn the strict rule into a defeasible rule, $a \wedge b \wedge c \wedge d \wedge e \Rightarrow\sim f$, this might help. Δ itself becomes one big argument, and Δ defeats π_3 on account of it being more specific. But then Δ itself contains π_3 and so it is self defeating. Thus we are still left with $a, b, c, \pi_1, \pi_2, \pi_3$ as the winning arguments and the anomaly stands.

By the way, a well known rule of nonmonotonic logic is that if $a \vdash b$ monotonically then $a \mid\!\sim b$ nonmonotonically. So we can add/use the strict rules in our arguments.

We can add the axiom

$$\frac{(x_1, \ldots, x_n) \rightarrow x}{(x_1, \ldots, x_n) \Rightarrow x}$$

So why are we getting anomalies? The reason is not our particular definition of defeat or the way we write the rules or the like.

The reason is that we do not allow for joint attacks. *You will notice that some of the devices used in Example 2.1 can help here, but they are not methodological. We*

are getting anomalies because outputs of successful arguments can join together in the strict reasoning part to get a contradiction, but their sources (i.e. the defeasible arguments which output them) cannot join together in a joint attack. See Example 2.2 which is very similar to this example.

The difference now in comparison with Example 2.2 is that we have precise definitions for our notions of defeat etc. and so we can define joint attacks, change the underlying logic or take whatever methodological steps we need.

The simplest way to introduce joint attacks in our system without changing the definitions is to add the following rule axiom schema for any Δ

$$(x_1, \ldots, x_n) \Rightarrow \top$$

for any $x_1, \ldots, x_n$*, any n. Thus we would have the proofs*

$$\begin{array}{l} \eta_3 : (\pi_1, \pi_2, (d, e) \Rightarrow \top) \\ \eta_2 : (\pi_1, \pi_3, (d, f) \Rightarrow \top) \\ \eta_1 : (\pi_2, \pi_3, (e, f) \Rightarrow \top) \end{array}$$

$$\eta_0 : (\pi_1, \pi_2, \pi_3, (d, e, f) \Rightarrow \top$$

The argument η_0 *is inconsistent, and we ignore arguments like* $(a, a \Rightarrow \top)$ *or* $(a, a \Rightarrow d, (a, d) \Rightarrow \top)$*, which give nothing new.*

Since attacks and defeats are done by the output of the arguments, we get that η_i *attacks and is being attacked by* π_i*.*

The resulting network will need a Caminada labelling and not all π_i, η_i *will always be winning.*

The outputs of the various arguments are as follows:

$$\begin{array}{lcl} \text{output}(a) & = & \{a\} \\ \text{output}(b) & = & \{b\} \\ \text{output}(c) & = & \{c\} \\ \text{output}(\pi_1) & = & \{a, d\} \\ \text{output}(\pi_2) & = & \{b, e\} \\ \text{output}(\pi_3) & = & \{c, f\} \\ \text{output}(\eta_3) & = & \{a, b, d, e\} \\ \text{output}(\eta_2) & = & \{a, d, c, f\} \\ \text{output}(\eta_1) & = & \{b, e, c, f\} \\ \text{output}(\eta_0) & = & \{a, b, c, d, e, f, \sim f\}. \end{array}$$

Figure 8 shows the network (we ignore the arguments which give nothing new). Clearly, any Caminada labelling will choose one of the pairs $\{\eta_i, \pi_i\}$. The justified theory will be consistent!

Definition 3.10 (Consequence relation based on defeat) *We assume we allow joint attacks as suggested in Example 3.9. Let* Δ *be a consistent theory and let a be a literal. We define the notion of* $\Delta \mid\!\sim a$ *as follows:*

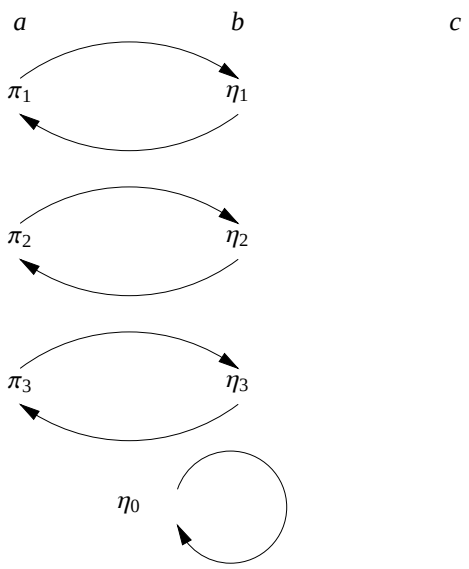


Figure 8:

Let $\mathfrak{A}$ be the set of all consistent arguments based on Δ and let $\mathcal{D}$ be the defeat relation as defined above. Then $(\mathfrak{A}, \mathcal{D})$ is a Dung framework. Let $\mathbb{T}$ be an admissible set of arguments (take some Caminada labelling or if you wish, take the unique grounded set) and let $\mathbb{A}_\Delta$ be the strict closure of the union of all outputs of the arguments in $\mathbb{T}$. Then we define

$$\Delta \mid\!\sim a \text{ iff } a \in \mathbb{Q}_\Delta.$$

Lemma 3.11 *$\mathbb{Q}$ is consistent.*

Proof. Otherwise we have several winning arguments. $\pi_i, i = 1, \ldots, n$ with $x_i \in \theta_\Delta(\pi_i)$ such that Δ^S and $\{x_i\}$ and the strict rules in Δ can prove y and $\sim y$. Assume n is minimal for giving a contradiction.

However, the argument

$$\eta_i = (\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n, (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \Rightarrow \top)$$

attacks and is being attacked by π_i.

So not all π_i can be winning! ■

Remark 3.12 *The exact results for $\mid\!\sim$ depend on the admissible set winning but the important point is that now the system is aware of the anomaly (inconsistency) and so we have no anomaly!*

To summarise, the devices we used are:

1. *joint attacks through the axiom $\bigwedge x_i \Rightarrow \top$*

2. *arguments attack through their output and not just through the head of the last rule in the argument. In other words, we always close under strict rules at every stage of the argument.*

Remark 3.13 (Failure of cut — 1) *This example shows that we cannot always chain proofs together.*

Let $\Delta = \{u, u \Rightarrow \sim b, \sim b \Rightarrow a, a \Rightarrow b\}$.

Then $\Delta \mid\!\sim a$ *because of* $\pi_a = (u, u \Rightarrow \sim b, \sim b \Rightarrow a)$.

We also have $\Delta, a \mid\!\sim b$ *because of* $\pi_b = (a, a \Rightarrow b)$. *However, we cannot string* π_a *and* π_b *together to get a proof for* $\Delta \mid\!\sim b$ *because* $(u, u \Rightarrow \sim b, \sim b \Rightarrow a, a \Rightarrow b)$ *is not consistent.*

Thus cut fails for the consequence relation of Definition 3.10. The next example shows failure of cut even when the proofs π_a *and* π_b *can consistently chain.*

Example 3.14 (Failure of cut — 2) *This is another example for the failure of cut for the consequence relation of Definition 3.10. Let* $\Delta = \{u, u \Rightarrow a, a \Rightarrow v, v \Rightarrow b, x, x \Rightarrow v, x \wedge v \Rightarrow w, x \wedge u \rightarrow \sim w\}$.

Then $\Delta \mid\!\sim a$ *because of*

$$\pi_a = (u, u \Rightarrow a).$$

$\Delta, a \mid\!\sim b$ *because of*

$$\pi_b = (a, a \Rightarrow v, v \Rightarrow b).$$

The outputs of π_a *and* π_b *together are* $\{u, a, v, b\}$ *and are consistent. So we can string the proofs together to* π_b^a *proving* b *from* Δ.

$$\pi_b^a = (u, u \Rightarrow a, a \Rightarrow v, v \Rightarrow b).$$

This proof however is defetated by the proof η *(which is consistent and undefeated).*

$$\eta = (x, x \Rightarrow v, x \wedge v \Rightarrow w).$$

The output of η *is* $\{x, v, w\}$. *The reason for the defeat is because*

1. *The head rule of* η *is more specific than that of* π_b^a.

2. *The union of the outputs of* η *and* π_b^a *is the set* $\{u, a, v, b, x, w\}$ *which is inconsistent because of the strict rule* $x \wedge u \rightarrow \sim w$. η *does not defeat* π_b *because we need* u *to get inconsistency.*

Example 3.15 (Success of cut) *Let* Δ *be the following database*

$$\Delta = \{u, x, a \Rightarrow v, v \Rightarrow b, x \Rightarrow \sim a, \sim a \Rightarrow v, u \Rightarrow x \wedge v \Rightarrow \sim b\}.$$

The arguments we can construct from $\Delta \cup \{a\}$ are as follows.

$$\begin{array}{rcl}
\pi_a & = & (u, u \Rightarrow a) \\
\pi_b & = & (a, a \Rightarrow v, v \Rightarrow b) \\
\eta & = & (x, x \Rightarrow \sim a, \sim a \Rightarrow v, x \wedge v \Rightarrow \sim b) \\
A_1 & = & (a, a \Rightarrow v) \\
A_2 & = & (x, x \Rightarrow \sim a) \\
A_3 & = & (x, x \Rightarrow \sim a, \sim a \Rightarrow v) \\
A_4 & = & (u, u \Rightarrow a, a \Rightarrow v) \\
\pi_b^a & = & (u, u \Rightarrow a, a \Rightarrow v, v \Rightarrow b) \\
B_1 & = & (a, x, a \Rightarrow v, x \wedge v \Rightarrow \sim b) \\
B_2 & = & (x, u, u \Rightarrow a, a \Rightarrow v, x \wedge v \Rightarrow \sim b).
\end{array}$$

We also have the atomic arguments $(a), (u)$ and (x). We have $\Delta \mid\!\sim a$ because of π_a and maybe $\Delta, a \mid\!\sim b$ because of π_b, but this is attacked and defeated by B_1. We now look at π_b^a and ask whether it is undefeated and hence shows that $\Delta \mid\!\sim b$. It is attacked by η and defeated.

4 Conclusion

We have proposed methodologically robust options for giving logical contents to nodes in abstract argumentation networks. We have provided a number of examples and considered a rigorous case study. We have also defined consequence relations based on a notion of defeat, considered rationality postulates, and proved that one such consequence relation is consistent. As future work we shall investigate the issue of network computation in connection with the general methodology of fibring, and the question of learning and adapting the network system further to new information and evolving scenarios so that statistical aspects of the data can also be taken into account by the logic. Our objective is to provide a unified theory of logic and network reasoning, from unifying principles to computational systems and applications. Along with [18] and [8], this paper is a step in this direction.

References

[1] C.E. Alchourron, P. Gardenfors and D.C. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions, The Journal of Symbolic Logic, 50: 510–530, 1985.

[2] H. Barringer, D. M. Gabbay and J. Woods. Temporal Dynamics of Support and Attack Networks: From Argumentation to Zoology. In D. Hutter and W. Stephan (eds), *Mechanising Mathematical Reasoning*, LNCS 2605: 59–98, Springer, 2005.

[3] P. Besnard and A. B. Hunter. *Elements of Argumentation*, MIT Press, 2008.

[4] M. W. A. Caminada and L. Amgoud. On the evaluation of argumentation formalisms, *Artificial Intelligence*, 171 (5–6): 286–310, 2007.

[5] A. S. d'Avila Garcez, K. Broda and D. M. Gabbay. Neural-Symbolic Learning Systems: Foundations and Applications, Springer, 2002.

[6] A. S. d'Avila Garcez, D. M. Gabbay and L. C. Lamb. Value-based Argumentation Frameworks as Neural-Symbolic Learning Systems. Journal of Logic and Computation 15(6):1041-1058, December 2005.

[7] A. S. d'Avila Garcez and D. M. Gabbay. Fibring Neural Networks. In Proc. 19th National Conference on Artificial Intelligence AAAI 2004. San Jose, California, USA, AAAI Press, July 2004.

[8] A. S. d'Avila Garcez, L. C. Lamb and D. M. Gabbay. Neural-Symbolic Cognitive Reasoning, Springer, 2008.

[9] D. M. Gabbay. *Labelled Deductive Systems*, OUP, 1996.

[10] D. M. Gabbay and U. Reyle. N-Prolog: An Extension of Prolog with Hypothetical Implications. Journal of Logic Programming, 1(4): 319-355, 1984.

[11] D.M. Gabbay. Fibring Logics. OUP, 1998.

[12] D. M. Gabbay and J. Woods. Resource origins of non-monotonicity. *Studia Logica*, 88 (1): 85–112, 2008.

[13] M. J. Gomez Lucero, C. I. Chesnevar and G. R. Simari. On the Accrual of Arguments in Defeasible Logic Programming. In Proc. 21st Intl. Joint Conference on Artificial Intelligence IJCAI 2009, Pasadena, USA, July 2009 (in press).

[14] H. Leitgeb. Neural network models of conditionals: an introduction. In X. Arrazola, J. M. Larrazabal et al. (eds.), Proc. ILCLI International Workshop on Logic and Philosophy of Knowledge, Communication and Action, 191-223, Bilbao, 2007.

[15] J. Pollock. Self-defeating Arguments. Minds and Machines, 1 (4): 367–392, 1991.

[16] B. Verheij. Accrual of arguments in defeasible argumentation. In Proc. 2nd Dutch/German Workshop on Nonmonotonic Reasoning, Utrecht, 217–224, 1995.

[17] H. Prakken and G. Sartor. Argument based extended logic programming with defeasible priorities. *Joural of Applied Non-classical Logics*, 7:25–75, 1997.

[18] K. Stenning and M. van Lambalgen. Human reasoning and cognitive science. MIT Press, 2008.