View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Open Repository and Bibliography - Luxembourg

arXiv:1504.07020v1 [cs.AI] 27 Apr 2015

Theory of Semi-Instantiation in Abstract Argumentation

D. M. Gabbay
Bar Ilan University, Israel
King's College London, UK
University of Luxembourg, Luxembourg
University of Manchester, UK
dov.gabbay@kcl.ac.uk*

Abstract

We study instantiated abstract argumentation frames of the form (S, R, I), where (S, R) is an abstract argumentation frame and where the arguments x of S are instantiated by $I(x)$ as well formed formulas of a well known logic, for example as Boolean formulas or as predicate logic formulas or as modal logic formulas. We use the method of conceptual analysis to derive the properties of our proposed system. We seek to define the notion of complete extensions for such systems and provide algorithms for finding such extensions. We further develop a theory of instantiation in the abstract, using the framework of Boolean attack formations and of conjunctive and disjunctive attacks. We discuss applications and compare critically with the existing related literature.

1 Motivation and Orientation

This paper studies semi-instantiated argumentation network of the form (S, R, I), where (S, R), $R \subseteq S \times S$ is an abstract argumentation network and I is an instantiation function, giving for each $x \in S$ a formula $I(x)$ of some logic **L**.

The attack relation R is not instantiated and remains abstract. We are not told, in terms of the logic **L**, why there is an attack.

There are several possibilities for such a system to arise.

Option 1. We can view such a system as semi-ASPIC like instantiation. The APPIC approach (see [32]) will start with a theory Δ in a logic, **L** define the notion of **L**-proofs for Δ and will elt S be the set of all possible such proofs and will further define the attack relation $R \subseteq S \times S$ in terms of relationships among these proofs. In our case we just take Δ as the arguments (no proof theory avaiable) and simply tell you abstractly what is supposed to attack what.

*Research supported by the Israel Science Foundation Project 1321/10: Integrating Logic and Networks.

For example, let $\Delta = \{A, A \to \bot\}$ (with $\to$ being implication, and $\bot$ being falsity) and the attack relation be from A to $A \to \bot$ but *not* from $A \to \bot$ to A. We are not explaining why this attack relation is defined so. There are logics, like the Lambek calculus where modus ponens works from the left but not from right. So $A, A \to B \vdash B$ holds, but $A \to B, A \nvdash B$. Thus if $X \twoheadrightarrow Y$ means $X, Y \vdash \bot$, then we get that $A \twoheadrightarrow (A \to \bot)$ holds ("$\twoheadrightarrow$" is attack) but $(A \to \bot)$ does not attack A.

Option 2. Such instantiated systems can also arise from general methodolgoical considerations. Let us ask ourselves a very simple question:

Question. *What is the added value of abstract argumentation networks over, say, classical propositional logic?*

Obviously they have the same expressive power. Many papers by various authors have demonstrated such equivalence. My favourite is my own paper [37], showing that the attack relation is really the Peirce–Quine dagger connective ($x \downarrow y =$ def. $\neg x \wedge \neg y$) of classical logic. So, we ask, what is the added value of such networks? My answer to this is that in these networks we bring some meta-level features into the object level Dung argumentation networks, expand classical propositional logic with the meta-predicate "x is false". When we write $z \twoheadrightarrow x$ (i.e., z attacks x) we are saying $z =$ "x is false", or $z \leftrightarrow$ "x is false" So the liar paradox becomes $x \twoheadrightarrow x$, "I am false". So the added value of abstract argumentation networks to classical propositional logic is the meta-predicate False(x). So the language now has $(\neg x, x \wedge y, x \vee y, x \to y)$ and the additional connective False (x).

Now the minute we accept this view we must also allow and address expressions like

$$x \twoheadrightarrow A$$

where A is a wff, i.e., $x =$ "A is false" and we then must also allow

$$x \wedge y \twoheadrightarrow A$$

meaning that x and y together say that A is false, and now, of course, once we go this far we must also address

$$B \twoheadrightarrow A.$$

The latter is nothing but the equivalence

$$B \leftrightarrow \text{ False}(A).$$

If you think about it, once we add to any logic a new connective "$C(x)$", we must be able to address $y \leftrightarrow C(x)$, it being just another wff.

Having established some interest in semi-instantiated argumentation networks, let us now get to business and describe the machinery and problems involved.

Let (S, R) be an abstract argumentation frame. This means that S is a non-empty set and $R \subseteq S \times S$.

Let $\mathbf{L}$ be a logic, with a set of well formed formulas WFF($\mathbf{L}$) and let μ be either semantics or proof theory for this logic. Assume that we have models for this logic

which we denote by $\{\mathbf{m}\}$, and/or theories for this logic which we denote by $\{\Delta\}$. We assume that a notion $\Vdash_{\mathbf{L}}$ for this logic is available such that for each $\mathbf{m}$ or Δ and for each $\Phi \in \text{WFF}(\mathbf{L})$ the relation $\Delta \Vdash_{\mathbf{L}} \Phi$ or $\mathbf{m} \Vdash_{\mathbf{L}} \Phi$ can get 3 answers. Yes (= 1), no (= 0) or undecided (= $\frac{1}{2}$).

As an example of a logic let us take intuitionisitc propositional logic **H**, with consequence $\vdash_{\mathbf{H}}$ we can have:

$$\begin{array}{l}
\Delta \Vdash_{\mathbf{H}} \Phi \text{ is 1 if } \Delta \vdash_{\mathbf{H}} \Phi \text{ holds} \\
\Delta \Vdash_{\mathbf{H}} \Phi \text{ is 0 if } \Delta \vdash_{\mathbf{H}} \neg\Phi \text{ holds} \\
\Delta \Vdash_{\mathbf{H}} \Phi \text{ is } \frac{1}{2} \text{ if neither } \Delta \vdash_{\mathbf{H}} \Phi \text{ or } \Delta \vdash_{\mathbf{H}} \neg\Phi \text{ holds}
\end{array}$$

Similarly if $\mathbf{m}$ is a propositional Kripke model for **H**, we can have

$$\begin{array}{l}
\mathbf{m} \Vdash_{\mathbf{H}} \Phi \text{ is 1 if } \mathbf{m} \Vdash \Phi \text{ holds} \\
\mathbf{m} \Vdash_{\mathbf{H}} \Phi \text{ is 0 if } \mathbf{m} \Vdash \neg\Phi \text{ holds} \\
\mathbf{m} \Vdash_{\mathbf{H}} \Phi \text{ is } \frac{1}{2} \text{ if neither holds}
\end{array}$$

Another example is 3-valued classical propositional logic with the Kleene truth table for $\{0, \frac{1}{2}, 1\}$. Call it **K**. See [19] and Definition C.2.

Given a model assignment $\mathbf{m}$ to the atoms, we have $\mathbf{m} \Vdash_{\mathbf{K}} \Phi$ is the value that $\mathbf{m}$ gives to Φ, denoted by $\mathbf{m}(\Phi)$. It is a value in $\{0, 1, \frac{1}{2}\}$.

Let (S, R) be a network and let **L** be a logic with $\Vdash_{\mathbf{L}}$. Consider the instantiation function, $I : S \mapsto \text{WFF}(\mathbf{L})$.

Consider (S, R, I). This is an instantiated argumentation network. We seek to define the notion of complete extensions for (S, R, I) and give algoirthms for finding such extensions.

After performing a conceptual analysis of this problem, we reached the following definition. A model $\mathbf{m}$ (or a theory Δ) of **L** generates an extension for (S, R, I) if the function $\lambda_{\mathbf{m}}$ (or λ_{Δ}) defined on S below is a legitimate Caminada labelling giving rise to a complete extension on (S, R).

The function λ is:

$$\begin{array}{l}
\lambda_{\mathbf{m}}(x) = \text{value of } (\mathbf{m} \Vdash_{\mathbf{L}} I(x)) \\
\lambda_{\Delta}(x) = \text{value of } (\Delta \Vdash_{\mathbf{L}} I(x))
\end{array}$$

The problem is how to identify/compute, using purely argumentation methods, such extensions for (S, R, I). This is the task of this paper.[1] Note that the emphasis is on

[1]The reader should note that we are not defining, as a stipulated technical definition, the complete extensions of (S, R, I) as those legitimate Caminada labellings arising from models or theories of the logic. We are deriving this definition from conceptual analysis of the idea of instantiation.

To make the point absolutely clear, suppose we instantiate the elements of S by names of Chinese restaurants in London. We can define by stipulation extensions for such Chinese systems as those legitimate Caminada extensions λ such that if

- $\lambda(x)$ = in then the Chinese restaurant associated with x made a profit in 2014
- $\lambda(x)$ = out, then the Chinese restaurant associated with x made a loss in 2014
- $\lambda(x)$ = undecided, then the Chinese restaurant associated with x came out even in 2014

The above stipulation has nothing to do with a Chinese restaurant attacking another, and is nothing more than means of restricting the Caminada labellings on (S, R).

using geometric syntactical argumentation methods to find the extensions of (S, R, I). What we can do and we do not want to do is to systematically generate all models $\mathbf{m}$ of the logic or all theories Δ of the logic and check whether $\lambda_{\mathbf{m}}$ or λ_{Δ} generate a legitimate Caminada extension. We want to syntactically transform (S, R, I) into an argumentation network. Put differently, we want to identify and use the argumentation network meaning of the logic.

We consider three main logics.

1. Classical propositional logic based on 3 valued Kleene truth table.
2. Monadic predicate logic without equality based on Kleene table.
3. Modal logic S5 based on Kleene table.

This paper solves the problem. However, many of the results are technical and are done in the Appendices.

The methodological schema is simple:

Given (S, R, I) with I being an instantiation into WFF($\mathbf{L}$) we follow the steps below:

Step 1. Rewrite any wff Φ of $\mathbf{L}$ into an equivalent formula (in $\mathbf{L}$) which is argumentation friendly and convenient form.

Finding the right friendly form is not immediate and requires some analysis and trial and error. Once we find a convenient form for any $\Phi \in$ WFF($\mathbf{L}$) we need to prove the equivalence. This may involve some technical manipulations and is therefore done in an appendix.

Step 2. The instantiation $I(x) = \Phi_x$ can now be assumed to be in this special form. When x attacks y in (S, R), we get after instantiation that Φ_x attacks Φ_y. This attack between formulas of $\mathbf{L}$, needs to be given a meaning. The two formulas may be consistent in $\mathbf{L}$, so what does it mean that one attacks the other? So we transform any Φ of $\mathbf{L}$ into an argumentation template called a Boolean attack formation "equivalent" to Φ which we denote by $\mathbb{BF}(\Phi)$. Such formations can attack each other because they are defined as argumentation systems with input output nodes.

Note that we *do not* want to say something like "Φ attacks Ψ if $\{\Phi, \Psi\}$ is not consistent in $\mathbf{L}$" because we do not want to use $\mathbf{L}$. We turn Φ and Ψ syntactically into argumentation networks and remain solidly within the argumentation framework world.

We now have $\mathbb{BF}(\Phi_x)$ attacking $\mathbb{BF}(\Phi_y)$ where each $\mathbb{BF}$ is an argumentation network with input and output nodes.

The attack formation associated with Φ, encodes the logical meaning of Φ. This has to be defined and proved. Because of the technical complexity of the transformation from Φ to $\mathbb{BF}(\Phi)$, this is also done in an appendix.

Step 3. We now have the original (S, R) network instantiated by Boolean attack formations. So this is a system of network of networks. Thus steps 1 and 2 reduced the problem of instantiating a network (S, R) by wffs of a logic $\mathbf{L}$, into the problem of instantiating (S, R) by some special argumentation networks called $\mathbb{BF}$ (Boolean attack

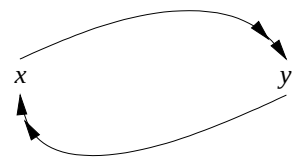


Figure 1:

formations) derived for **L**. Such a purely argumentation problem is of interest in its own right. We have a complex system (S, R, I), where (S, R) is an argumentation network and for each $x \in S, I(x) = (S_x, R_x)$ is an argumentation network.

We seek to turn this system into one big master argumentation network (S^M, R^M). We now want to define the notion of complete extensions for this network (S^M, R^M).

Remember that this network is not arbitrary but was constructed to do the job of finding extensions for instantiations of (S, R) into WFF(**L**).

We find by conceptual analysis that the traditional notions of extensions for neworks is not adequate for the job and that we need to develop and motivate our own new notion of extension (which we call *non-toxic truth intervention extensions*).

This new notion of extensions is, in fact, a paper in its own right and because of its complexity it is done in an appendix.

Step 4. We now have (S^M, R^M) and a new method of finding extensions for it. We generate these extensions and from the extensions we generate models for the logic **L**. The models for **L** give us extensions for (S, R, I), with I an instantiation into WFF(**L**).

The above step by step workflow schema is quite simple but is rather technical. So most of the work is done in the appendices so as not to burden the reader.

It would be of value to illustrate the steps of the workflow on an example.

Start with (S, R) being a two point loop $\{x, y\}$ with x attacking y and y attacking x. See Figure 1.

We have three extensions:

$E_1 : x =$ "in", $y =$ "out"
$E_2 : x =$ "out", $y =$ "in"
$E_3 : x = y =$ "undecided"

Instantiate with

$I(x) =$ someone is tall
$I(y) =$ Dov is tall

We get Figure 2.

We are looking for (Kleene three valued) predicate models which would give values to $\exists x T(x)$ and $T(d)$ which will form an extension.

Our knowledge of logic tells us that there is only a model for $\exists x T(x) \wedge \neg T(d)$, e.g. Dov and Lydia, with Dov not tall and Lydia tall. There is no model with $T(d) \wedge \neg \exists x T(x)$. Thus our knowledge of logic tells us that the above instantiation has only

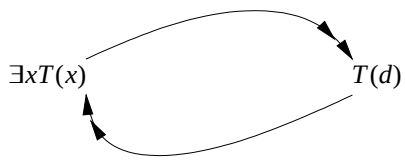


Figure 2:

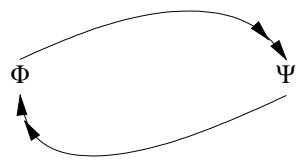


Figure 3:

one extension in classical two valued monadic logic. In Kleene 3 valued logic, we can also have a model with $\exists x T(x) = T(d) = \frac{1}{2}$.

However, we are not supposed to use our knowledge of logic but we must use an algorithm to find the model. So as far as the algorithm is concerned we have in this example a case of the instantiation of Figure 3

We turn Φ and Ψ into equivalent networks of Boolean attack formation which we draw as in Figure 4 and Figure 3 becomes Figure 5 which is a master network representing the instantiation. We find extensiosn for the master network and these extensions will generate models.

The perceptive reader might ask why not look at $\Phi \wedge \neg\Psi$ and $\Psi \wedge \neg\Phi$ and find models for them? After all, we know the logic **L**? The answer is that we are not using the logic at all. The transformation Φ to $\mathbb{BF}(\Phi)$ is pure syntax. It is the argumentation extensions which find the models for $\{I(x) | x \in S\}$ which respect the constraints imposed by the original (S, R).

To continue analysing this example from the argumentation point of view, assume the universe has only two people, Dov and Lydia. So $\exists x T(x)$ becomes $T(d) \vee T(l)$.

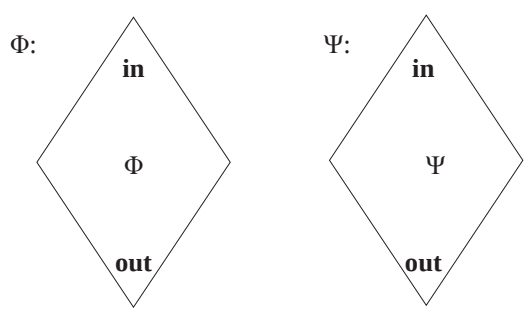


Figure 4:

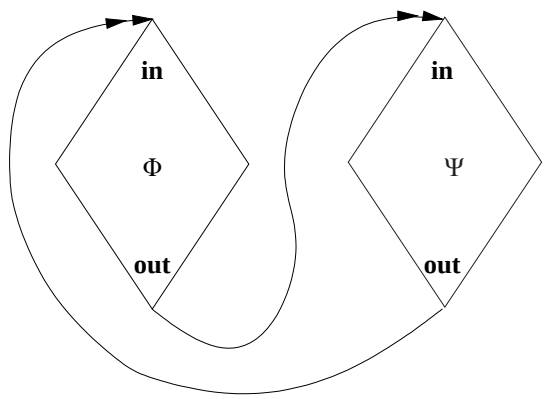


Figure 5:

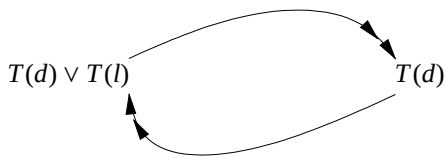


Figure 6:

Figure 2 becomes Figure 6.

The reader can see now the logic behind this instantiation. We have to define correctly what it means to attack a disjunction and what it means for a disjunction to mount an attack. Once we do that (this is done later in tis paper) we get that Figure 6 is equivalent to Figure 7

The only extensions are

1. $T(l) = \text{in}$, $T(d) = \text{out}$ (corresponding to $\exists x T(x) \wedge \neg T(d)$) and
2. $T(l) = T(d) = \text{undecided}$ (corresponding to $\exists x T(x) = T(d) = \frac{1}{2}$).

Note that we can use the above as an argumentation theorem prover to check the consistency of $\Phi \wedge \neg\Psi$ and $\neg\Phi \wedge \Psi$.[2]

Let us conclude this preliminary orientation by saying a few words about our deductive and expositional approach. Our approach is new. We arrive at our proposed system through a conceptual analysis (using common sense) of the components needed for an abstract theory of instantiation in general and for the specific instance of, for example, predicate and modal argumentation and following this analysis we define our system. So, as we are writing the present lines, we do not know yet the full details of what kind of system we will get.

[2]Existing machines for finding extensions for argumentation networks push the problem to logical provers. So we are not suggesting our reduction as a practical theorem prover for logic but only to highlight that we are operating purely in the argumentation world.

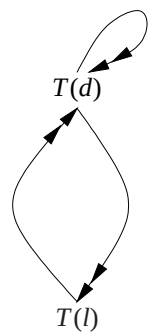


Figure 7:

We choose to deal, for the sake of simplicity, with the classical Boolean propositional calculus and with monadic classical predicate logic without equality and with modal logic S5. See Appendix A. We shall deal with more complex logics in a subsequent paper.

There exists in the literature the instantiated approach to argumentation, also known as ASPIC, see [13, 14, 16]. This approach is related but not the same as our approach. See Appendix E for full comparison and discussion.

1.1 Structure of our paper

Our program for this paper has the following methodological structure:

Starting point. We assume as our given starting point Dung theory of atomic finite propositional abstract argumentation frames. Namely frames of the form (S, R), where S a finite set of atomic arguments, $R \subseteq S \times S$ is the attack relation together with the traditional notion of complete extensions $E \subseteq S$ (to be recalled and defined in the next section).

Objective. Extend what is given to classical propositional calculus and to monadic predicate logic and to modal logic, namely allow the elements of S to be instantiated as formulas of classical propositional calculus or respectively as formulas of (monadic) predicate logic or respectively as formulas of modal logic S5 and define the concept of a complete extensions for this case in a natural and completely syntactical and combinatorial way, without using any logical notions.[3]

[3]By completely syntactical and combinatorial way we mean we use only the geometry of the graph (S, R) and possibly a new concepts of attack based on the geometry of the graph and on the simple syntactical structure of the arguments. So we assume a system of the form (S, R, I), where S is a set of atomic arguments, R is the attack relation, and I is an instantiation function giving for each $x \in S$ a formula $I(x)$ of propositional logic or of predicate logic or resp. modal logic. We look for extensions respecting the instantiation I, but such extensions are to be defined purely syntactically. So defining ARB iff $A = \neg B$ or $B = \neg A$ is acceptable but defining ARB iff $A, B \vdash \bot$ is not!

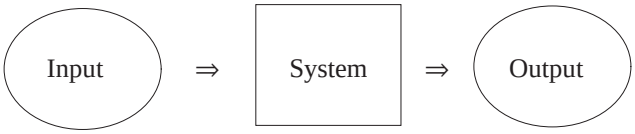


Figure 8:

Methodological approach. To achieve our objective we use the method of common sense conceptual analysis, a well known method in philosophy circles.

Conclusion. We produce an abstract theory of instantiation of argumentation frameworks in general and for the specific cases of Boolean, predicate, and modal instantiations, and compare and discuss what we get with systems proposed in the literature.

Let us now begin this rather unusual approach.

1.2 General methodological remarks

1. First, we remark that there are two major approaches to extending any propositional system to a predicate system, in this case extending propositional abstract argumentation to predicate or to modal S5 argumentation:

   *1. Look at applications areas of the (propositional) system and see its shortcomings and seek to extend it accordingly (to a predicate system or to a modal system). The needs of the applications will dictate what kind of generalisation to adopt.

   *2. Look formally at the (propositional) system and its components and use theoretical considerations to extend it by adding quantifiers or modalities.

   Personally we believe in the (*1) approach. In our particular case, however, the (*2) approach is just as good, because adding predicates and quantifiers or modalities to a propositional system is universally done for many logics. This is a well trodden path and it can lead to good extensions which will do well with applications, as long as our conceptual analysis is done with good common sense and care!

2. Second, let us recall a general methodological remedy for fixing any system which does not behave properly.

   Suppose we have an input/output system of some kind, as represented in Figure 8.

   Suppose the output is problematic and not to our liking. How do we remedy the situation? There are three pure traditional approaches and many options using their various combinations

   r1. Restrict the input to make sure the output is acceptable.

r2. Fix the system.

r3. Modify the output to make it acceptable.

To ilustrate, suppose we write a program for adding two numbers x and y, to get $x \oplus y$.

$$\{x, y\} \twoheadrightarrow x \oplus y.$$

Suppose we get the correct answer for

$$0 \leq x, y < 100,$$

but for $x \geq 100$ or $y \geq 100$ we get the answer

$$x \oplus y = x + y + 1$$

where "+" is the correct addition.

(r1) says do not use numbers ≥ 100.

(r2) says fix the program for $\oplus$

(r3) says subtract 1 from the result for the case that one of the input numbers is ≥ 100 and you will get the correct answer.

1.3 Conceptual analysis for instantiated Boolean or predicate argumentation

Let us first analyse some characteristics of the propositinal case and list their conceptual significance.

An abstract argumentation network [15] has the form (S, R), where $S \neq \emptyset$ is the set of abstract arguments and $R \subseteq S \times S$ is the attack relation. (We also write $x \twoheadrightarrow y$ to denote $(x, y) \in R$, reading x attacks y, especially in figures.)

The formal machinery associates with each (S, R) several types of extensions. It is convenient for us in this paper to use the Caminada labelling approach to extensions. See Caminada–Gabbay survey paper [11]. The Caminada labelling has the form $\lambda : S \mapsto \{1, 0, \frac{1}{2}\}$ where $\lambda(x) = 1$ means x is "in", $\lambda(x) = 0$ means x is "out" and $\lambda(x) = \frac{1}{2}$ means x is "undecided".

The exact definitions and background will be given in the next section in Definition 2.3. Here, in the orientation section, let us just note that because we regard the elements of S as atomic symbols, we can also view them as interpreted into any logic with $\neg$, as atomic propositions of that logic, and the set

$$T_\lambda = \{q | \lambda(q) = 1\} \cup \{\neg q | \lambda(q) = 0\}$$

is always consistent in any such logic, giving us no logical problems whatsoever.

This fact makes any coherent process giving rise to a λ and T_λ an acceptable process.

Let us assume we have a process for finding such acceptable functions λ. Our theoretical considerations for extending the propositional case to a predicate or modal extension can follow several tracks.

Track (t1): Substitution. Substitute for elements of S predicate wffs and apply the process and see what happens and when in difficulty offer suitable remedies.

Track (t2): Translation. Look at translations of the propositional theory into other predicate systems (translation into logic programming or into classical logic or into modal logic) and see the behaviour of the image of the source in the context of the target and find out how to import predicate logic argumentation or modal argumentation from the translation.

At this point of our deliberations, Track (t2) seems more difficult than Track (t1). Let us therefore begin with (t1).

1.4 The substitution track

Problems arise when we instantiate the elements of S and give them internal structure, such as a wff of predicate logic. The elements of T_λ, when having an internal structure, may clash with one another and render T_λ inconsistent, in whatever logic we happen to be using.[4] This means that the simplistic approach of allowing propositional or predicate wffs to be substituted for the atoms in S will most likely be problematic and may require a remedy. As part of our conceptual analysis we will proceed along this path and seek to identify possible remedies.

Even the simplest possible instantiation can be problematic. Suppose we take an argumentation network (S, R) and choose a single y in S and instantiate just this single y as the propositional constant $\top$. What happens? $\top$ is always true, so this is equivalent to saying that y should always be "in". This means that any x attacking y should be "out". This may sounds simple but it is not, because it changes the rules of the game: Firstly there may not be extensions where y is "in". So we have to say that we are dealing with networks which could be without semantics. Secondly the direction of attacks is no longer only following the arrows, but if there is an arrow leading onto $\top$, the attack is directed opposite the direction of the arrows. Dealing with such instantiation looks simple but it is not and so we postpone any further discussion to Appendix B.

In this section we want to illustrate our conceptual analysis on a less subtle case, so we deal with predicate instantiation.

Figures 9a and 9b present us with a simple network and a predicate instantiation for it, to serve as a simple example for our conceptual analysis:

The extension λ for Figure 9a is $\lambda_a(x) = \lambda_a(z) = 1$ and $\lambda_a(y) = 0$.

---

[4] Let

$$E^1_\lambda = \{q | \lambda(q) = 1\}$$

$$E^0_\lambda = \{q | \lambda(q) = 0\}.$$

These two sets are disjoint and therefore when we interpret the union of these two sets in a logic via the special instantiation

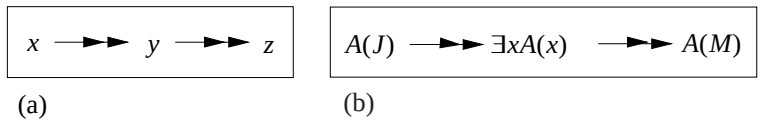
$$I(q) = q, \text{ for } q \in E^1_\lambda$$

and

$$I(q) = \neg q, \text{ for } q \in E^0_\lambda$$

to form T_λ, we get a consistent set in the logic.

However for any other possible interpretation/instantiation I^*, we may get inconsistency in the logic.



(a) (b)

Figure 9:

Thus $T_{\lambda_a} = \{x, \neg y, z\}$.

Upon instantiation in Figure 9b we get

$$T_{\lambda_b} = \{A(J), \neg\exists x A(x), A(M)\}.$$

This theory is inconsistent. Here A is a unary predicate and J and M are elements in the predicate universe.

We now recall the general methodological remedies (r1)–(r3) disucssed in subsection 1.2.

We have here an input output system

Inputs: Instantiated (S, R) with predicate formulas

Outputs: Predicate extensions

Problem: The output may be inconsistent.

The systems needs a remedy.

The (r1) option would restrict the input, maybe to some fragment of predicate logic. This is a very simple case, there is nothing to restrict.

The (r2) option would mean that we change the process of finding extensions.

The (r3) option means that we somehow revise the result and render it consistent.

For example we can use an idea of Sanjay Modgil from his PhD thesis and regard the output predicate theory T_λ as a theory in a defeasible predicate logic. Thus $\neg\exists x A(x)$ is a defeasible rule with $A(J)$ and $A(M)$ as exceptions. However, we do want T_λ to be in classical logic, so (r3) is not an option for us.

It seems the simplest remedy for us is to use (r2). Revise the process.

Seeking a remedy, let us simplify and assume that our predicate universe contains only two elements J and M and see whether this simplifications helps us get some ideas. Thus we have

- $\exists x A(x) \equiv A(J) \vee A(M)$
- $\forall x A(x) \equiv A(J) \wedge A(M)$.

Figures 9b becomes Figure 10

To continue our conceptual analysis we need to deal with, and seek a remedy for, the problems of Figure 10. In other words we need to solve the problem of instantiation into the classical propositional calculus, where $I(x)$ can give an arbitrary formula of classical propositional logic as values for x.

$$A(J) \twoheadrightarrow A(J) \vee A(M) \twoheadrightarrow A(M)$$

Figure 10:

Well, looking at Figure 8, let us give meaning to attacks on disjunctions and a meaning to disjunctions attacking other elements.

The case of x attacking the disjunction $y \vee z$ is complicated to express and it will be dealt with in Appendix C. Basically we have to express the Boolean equation $\neg x \leftrightarrow (y \vee z)$, and for this we need disjunctive attacks.

The case of a disjunction $y \vee z$ attacking x is simple to express.

- $y \vee z \twoheadrightarrow x$ means $y \twoheadrightarrow x$ and $z \twoheadrightarrow x$.

The reason being the meaning of $a \twoheadrightarrow b$. It means

- if a = "in", then b = "out".

So

- if $y \vee z$ is "in" then x is "out"

is equivalent to

- if [y is "in" or z is "in"] then x is "out"

which is equivalent to the conjunction of [if y is "in" then x is "out"] and [if z is "in" then x is "out"].

Thus Figure 10 becomes Figure 11, which is the same as Figure 12

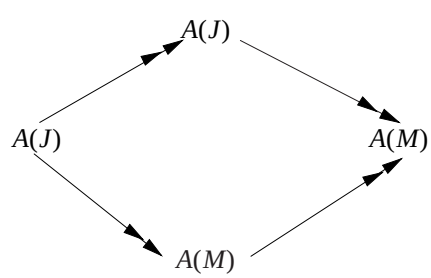


Figure 11:

The need for attacks from and attacks to conjunctions and disjunctions of atoms has already been considered by us in 2009 in connection with fibring argumentation networks. Figure 13 explains our notation from 2009 (full details are given in Appendix C).

It appears that we may now have a plan for a remedy of how to deal with monadic predicate logic substitutions $I(x)$ for nodes x in argumentation networks:

1. Eliminate the quantifiers in terms of conjunctions and disjunctions

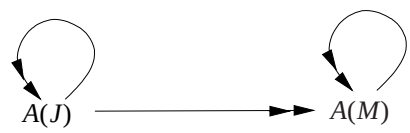
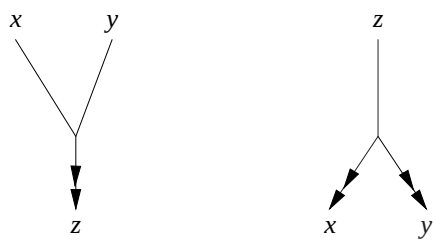


Figure 12:



(a) (b)

(a) x, y jointly attack z. If $x = y = 1$ then $z = 0$.

(b) z disjunctively attacks x, y. If $z = 1$ then either $x = 0$ or $y = 0$ or both equal 0.

Figure 13:

2. Develop a theory of attacks involving disjunctions and conjunctions.

However, the reduction of the quantifiers to disjunctions and conjunctions and the (yet to be described) argumentation networks with joint and disjunctive attacks does not solve our problem. Even for finite models the number of elements is not bounded and so we cannot replace $\exists x A(x)$ by a finite disjunction. We can attempt to say use a closed world assumption and use all the names mentioned explicitly in the network. This may work but not easily. We may have $J_1, \ldots, J_k$ as all the names with $A(J_i), i = 1, \ldots, k$ being "in", but nevertheless $\exists x A(x)$ being out. Worse still, if we allow predicate formulas for the form $\forall x \exists y \Phi(x, y)$, we may be forced to have an infinite number of elements. So this is not the way to go, at least not as a first attempt at the problem.

Looking again at Figure 9b, we suddenly make a surprising realisation. $A(J)$ is being essentially attacked by $\exists x A(x)$ when $\exists x A(x)$ is "out"!

We would expect, as in the case of propositional networks, that when an argument is out, then it is "dead". It has no effect. In the case of $\exists x A(x)$ when it is out it has an effect. This means, when taken to its full conclusions, that being "in", "out", "undecided" is not a value but it is a *state*, from which an argument can mount attacks. Figure 14 illustrates this new point of view.

In fact, to be completely coherent, we must allow for attacks of the form

$$(a \text{ is in state } \xi_1) \twoheadrightarrow (b \text{ is in state } \xi_2)$$

This brings us to the idea of what we call *state argumentation networks*, a new

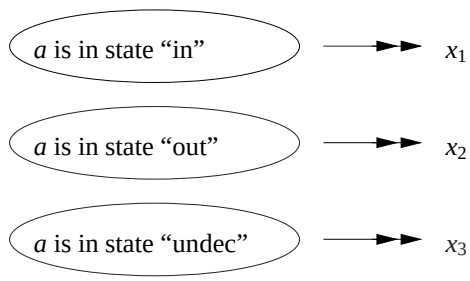


Figure 14:

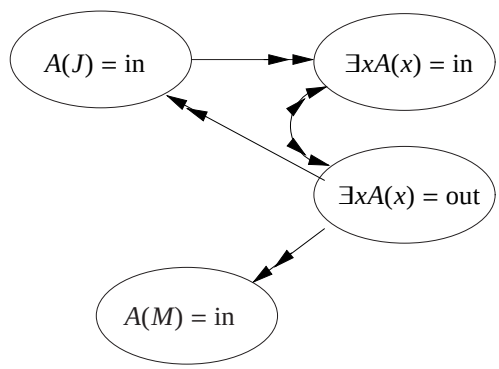


Figure 15:

concept, which once made precise, can help us represent our original goal, that of predicate argumentation nework. We can possibly transform Figure 8b into Figure 15.

A reader might say why not use negation as in Figure 16?

We can do that, but in general, an argument can have more than two states. State argumentation networks is a more general concept and we may wish to continue and develop it in this paper. Let us define it intuitively.

Definition 1.1 *Let* (S, R) *be a network and assume that* $S = S_1 \cup \ldots \cup S_k$ *with* $S_i \neq 0$ *and* $S_i \cap S_j = \emptyset, i \neq j$*. Also assume that for each* i *and each* $x, y \in S_i$*, such that* $x \neq y$ *we have that* xRy *hold. Under these conditions we can regard* (S, R) *as a state*

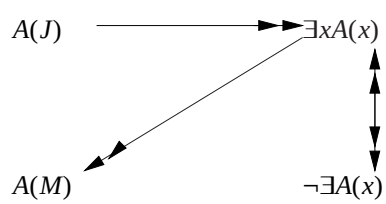


Figure 16:

argumentation network, where the elements are $\{S_i\}$ and each $x \in S_i$ is a different state of S_i.

Figure 9b can become Figure 17. $\top$ is attacking all $\neg x$ nodes where x is not attacked in the original figure.

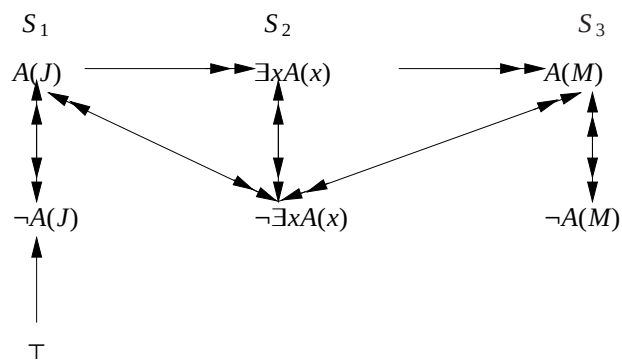


Figure 17:

Definition 1.2 *A two state argumentation network has the form*

$$(S \cup S^{\neg} \cup \{\top\}, R)$$

where S is a set of atoms, $S^{\neg} = \{\neg q | q \in S\}$ and $\top$ is top. We have

- $\neg \exists x (xR\top)$
- $\forall x (xR\neg x)$
- $\forall x (\neg xRx)$

Lemma 1.3 *Every (S, R) is equivlaent to (S^*, R^*) where $S^* = S \cup S^{\neg} \cup \{\top\}$ and R^* is*

$$R \cup \{(\top, \neg x) \mid x \in S\} \cup \{(x, \neg x), (\neg x, x)\}.$$

The extensions E^* of (S^*, R^*) are exactlly the extension E of (S, R) augmented by $\{\top\}$.

Proof Start with (S, R). Create $S^{\neg} = \{\neg x | x \in S\}$ and assume $\top \notin S$. Let $S^* = S \cup S^{\neg} \cup \{\top\}$. Let R^* be defined as $R^* = R \cup \{(x, \neg x), (\neg x, x) | x \in S\} \cup \{(\top, \neg x | x \in S\}$. It is clear that $\top$ in S^* attacks all the new points of $S^{\neg}$ which we added. Thus any extension E of (S, R) becomes the extension $E \cup \{\top\}$ of (S^*, R^*) and vice versae. ■

1.5 Summary of our plan so far for monadic predicate instantiation

We propose, at this stage of our conceptual analysis, the following plan.

We are given an abstract argumentation network (S, R), with an instantiation function I, giving for each $x \in S$ a formula of monadic predicate logic. We want to deal with it.

a. First we prove some theorems that the input from predicate logic can be restricted, without loss of generality, to an argumentation friendly form.

b. Assuming the input is of this form, we use its syntactical form together with R, to move to a new abstract argumentation network (S^*, R^*, I^*), with S a subset of S^* and R a subset of R^*.

   We take extensions E^* for the new network and look at $E^* \cap S$. We declare these as the sought for extensions for the original (S, R, I).

c. Hopefully we will prove that $\{I(x) \mid x \in E^* \cap S\}$ is consistent.

To achieve this we need some technical results.

The following is the list:

1. Define the notion of a 2-state argumentation network. Show that such networks are a special case of abstract argumentation network in the sense that they can be identified by special properties on the attack relation R.

2. Show that every argumentation network (S, R) can be embedded in a larger 2 state argumentation network (S^*, R^*) in a critical way. This means that (S, R) preserves all it properties even though it is part of the larger network. Thus we can say that every argumentation network is equivalent to a 2-state argumentation network. The equivalence is shown by a linear general transformation.

3. Show that every formula Φ of monadic predicate logic is classically equivalent to a formula $\bar{\Phi}$ in a standard argumentation friendly form, to be defined and to be convenient for our objective.

4. For every ordinary (S, R), define a 2-state (S^*, R^*) called the associate of (S, R), by

$$S^* = S \cup S^{\neg} \cup \{\top\}$$
$$R^* = \{(x, \neg x), (\neg x, x) \mid x \in S\} \cup \{\top, y \mid y \text{ not attacked in } (S, R)\} \cup R$$

   This is not the embedding described in (2) above.

5. Given an (S, R) form the associate (S^*, R^*). In order to instantiate (S, R) with predicate formulas $x \mapsto \Phi_x$, for $x \in S$, use instead (S^*, R^*) of (4) above and instantiate the nodes with standard form formulas, $x \mapsto \bar{\Phi}_x$ and $\neg x \to \neg\bar{\Phi}_x$.

   Let $I(z)$ for $z \in S^*$ be the instantiation function. We look at (S^*, R^*, I) and rewrite (transform) the network in an easy and purely syntactical way into a new network (S^*, R^{**}, I) and then take extensions. In this way, we hope, the correct consistent extensions are obtained. Thus the extensions E^{**} thus obtained restricted to S shall be declared as the predicate extensions of (S, R, I).

It is useful to give an example.

Example 1.4

1. *Start with the network* (S, R) *of Figure 9a.*

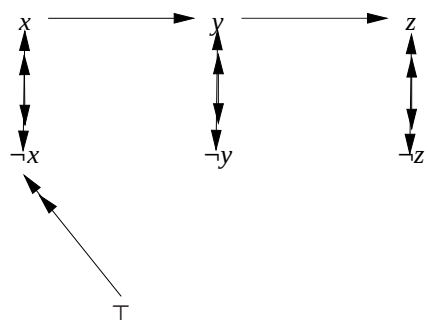


Figure 18:

2. *Transform it to the equivalent Figure 18.*

3. *Substitute/instantiate:*

$$I(x) = A(J)$$
$$I(y) = \exists x A(x)$$
$$I(z) = A(M)$$

and adjust the figure by adding attacks from any $\neg\exists x P(x)$ *onto any* $P(y)$ *and from any* $\forall x P(x)$ *onto any* $\neg P(y)$.

Note that this is done purely syntactically without taking into consideration any logical meaning of the instantiations formulas. We get Figure 17.

4. *Calculate traditional extensions. We get the extensions*

$$\{\neg A(M), \neg\exists x A(x)\}$$

We declare these as the extensions of (S, R, I).

Remark 1.5 *The perceptive reader looking at the extensions obtained in Example 1.4 for the network of Figure 9b might justifiably ask, what is the intuition behind this? Our answer is that in this case there we should not expect much intuition. We took an arbitrary abstract network coming from nowhere and substituted arbitrary predicate formulas into it. What kind of result would you expect, beyond that it is consistent? Nevertheless, let us look at the result from an AGM revision point of view. Our original theory was* $\{A(J), \neg\exists x A(x), A(M)\}$ *and we offer the revision option*

$$\{\neg A(M), \neg\exists x A(x)\}.$$

This is a maximal consistent sub-theory, and it makes sense if priority is given to $\neg\exists x A(x)$.

The real test for our intuition, however, in the case where we take a set of arbitrary predicate formulas, is to regard them as a network (i.e. with the empty attack domain) and apply the process to them. Do we get all maximally consistent subsets as the family of all extensions? This is the real test.

To be more precise, let our starting network (S, R, I) be with R empty, i.e. no attacks, and apply our process to it. See what we get. This is the intuitiveness test.[5]

Discussion and comparison with the literature will follow in the appropriate later section.

2 Abstract instantiated argumentation frames (AIAF)

This is a more formal section which will deal with several types of argumentation frames where the abstract arguments are instantiated by formulas of some logic. We consider classical propositional logic, classical monadic predicate logic without equality and modal logic S5. We also discuss other possibilitites such as instantiating with Boolean Attack Formations (BAFs, see Appendix C2). We examine our options, then propose a more general theory, and compare with the literature.

2.1 Instantiating with formulas of propositional logic

This sub-section is a case study, leading to to the next subsection 2.2, which will give concrete definitions.

Definition 2.1 *The classical propositional calculus is built up syntactically as follows:*

1. *A set Q of atomic propositions.*
2. *The classical connectives $\{\neg, \wedge, \vee, \rightarrow\}$, which are used to define the traditional notion of a formula, (wff).*
3. *The traditional notion of "the formula Φ of classical propositional logic is consistent", defined either semantically or proof-theoretically. We do not care how it is done. We just need to use it.*

Definition 2.2

1. *An abstract argumentation frame has the form (S, R), where S is a set of atomic symbols (we use for S distinct symbols from those we use for Q of Definition 2.1), and $R \subseteq S \times S$ is the attack relation. We also denote xRy by $x \twoheadrightarrow y$.*
2. *We follow Dung [15] and define the notion of complete extension $E \subseteq S$ as a set satisfying the following:*
   (a) *E is* conflict free, *namely for no $x, y \in E$ do we have xRy.*
   (b) *E* protects *its members, where "E protects x" means that $\forall z(zRx \rightarrow \exists y \in E(yRz))$.*
   (c) *E is* complete, *namely if E protects x then $x \in E$, for any $x \in S$.*

[5] This type of network (i.e. arbitrary S, empty attack relation and any instantiation into classical propositional calculus) can be viewed as an Abstract Dialectical Frame of Brewka and Woltran, see [2].

3. *It is well known that complete extensions always exist, though they might be empty.*

Definition 2.3 *Let (S, R) be an argumentation frame. Let $\lambda : S \mapsto \{0, \frac{1}{2}, 1\}$ be a labelling function. We say λ is a legitimate Caminada labelling iff the following holds:*

1. *If $\neg\exists y(yRx)$ then $\lambda(x) = 1$*
2. *If for all y s.t. yRx we have $\lambda(y) = 0$, then $\lambda(x) = 1$.*
3. *If for some y such that yRx we have $\lambda(y) = 1$ then $\lambda(x) = 0$.*
4. *If (a) for all y such that uRx we have $\lambda(y) < 1$ and (b) for some y such that yRx we have $\lambda(y) = \frac{1}{2}$ then $\lambda(x) = \frac{1}{2}$.*

Lemma 2.4 *Let (S, R) be an argumentation network. Let E be a complete extension as defined in Definition 2.2. Let λ_E be defined by*

$$\lambda_E(x) = \begin{cases} 1 \text{ if } x \in E \\ 0 \text{ if } \exists y \in E yRx \\ \frac{1}{2} \text{ otherwise} \end{cases}$$

Then λ_E is a legitimate Caminada labelling.

Proof

1. If $\neg\exists y(yRx)$ then x is in E and hence $\lambda_E(x) = 1$.
2. If for all y such that (yRx) we have $\lambda_E(y) = 0$ then for all such y there is a z such that zRy and $z \in E$. Thus E protects x, hence $x \in E$, hence $\lambda(x) = 1$.
3. If for some y such that yRx we have $\lambda_E(y) = 1$, then $y \in E$ and hence $\lambda_E(x) = 0$ by definition.
4. Assume

   (a) for all y such that yRx we have $\lambda_E(y) < 1$

   (b) for some y_0, y_0Rx we have $\lambda_E(y_0) = \frac{1}{2}$.

   We want to show that $\lambda(x) = \frac{1}{2}$.

   From (a) we get that $\neg\exists y \in E(yRx)$. Thus $\lambda_E(x) \neq 0$.

   We show that $\lambda_E(x) \neq 1$, i.e. $x \notin E$. From (b) above, y_0Rx and $\lambda(y_0) = \frac{1}{2}$. Hence $\neg\exists z \in E(zRy_0)$. This means that x is not protected by E and hence $x \notin E$.

■

Lemma 2.5 *Let λ be a legitimate Caminada labelling and let $E_\lambda = \{x|\lambda(x) = 1\}$, then E_λ is a complete extension.*

Proof We show E_λ is a complete extension:

1. If $\neg\exists y(xRy)$ then $\lambda(x) = 1$ and so $x \in E_\lambda$.

2. E_λ is conflict fee because if $\lambda(x) = 1$ and xRy then $\lambda(y) = 0$ and $y \notin E_\lambda$.

3. If $\lambda(x) = 1$ and yRx then $\lambda(y) < 1$. We must also have for such y that $\lambda(y) = 0$ for otherwise we would get $\lambda(x) = \frac{1}{2}$. But $\lambda(y) = 0$ for all such y means that $\forall y(yRx \rightarrow \exists z(zRy \wedge \lambda(z) = 1)$. This means that E_λ protects all of its members.

4. Suppose E_λ protects x. This means that E_λ attacks all of the attackers of x. This means $\forall y(yRx \rightarrow \lambda(y) = 0)$ Therefore $\lambda(x) = 1$ and so $x \in E_\lambda$.

■

Definition 2.6 *Let (S_i, R_i) for $i = 1, 2$ be argumentation frames. Let $I_{1,2}$ be a function from S_1 to S_2. We form the network $(S_{1,2}, R_{1,2})$ called* the abstract instantiation of (S_1, R_1) by (S_2, R_2) using $I_{1,2}$ *as follows:*

$$\begin{array}{lcl} S_{1,2} & = & \{I_{1,2}(x) | x \in S_1\} \\ R_{1,2} & = & \{(y,z) | \textit{for some } a, b \in S_1, I_{2,3}(a) = y \textit{ and } I_{1,2}(b) = z \\ & & \textit{and } (y,z) \in R_1\} \cup R_2 \restriction S_{1,2}. \end{array}$$

Example 2.7

1. *Let* **L** *be a logical system. Let Φ, Ψ be two wffs of* **L**. *Let $\rho(\Phi, \Psi)$ mean that Φ, when "added" to Ψ causes "incompatibility". (We are not saying "$\{\Phi, \Psi\}$ is inconsistent" because we do not say anything about the logic and ρ may not even be symmetrical. In some logics $\Phi \circledcirc \Psi$ is not the same as $\Psi \circledcirc \Phi$). Let WFF(***L***) be all the well formed formulas of the logic. We can consider the network (WFF(***L***), ρ)).*

2. *Given (S, R) and a logic* **L**, *we can instantiate it by (WFF(***L***),ρ) as defined in Definition 2.6. This will include classical logic instantiation.*

   For logics **L** *which have a negation symbol $\neg$ (e.g. classical modal, monadic, or intuitionistic logics) we can require (S, R) to be a 2-state network as in Definition 1.2 and require the instantiation function $I : S \mapsto WFF(\mathbf{L})$ to satisfy*

$$I(\neg x) = \neg I(x).$$

Definition 2.8

1. *An abstract instantiated Boolean argumentation frame $(B - AIAF)$ has the form (S, R, I) where (S, R) is an abstract argumentation frame and I is a function, giving for each $x \in S$, a formula $\Phi_x(\{q_1, \ldots, q_n\})$ of the classical propositional calculus.*

2. *We write*

$$I(x) = \Phi_x(\{q_1, \ldots, q_m\})$$

   indicating that the classical propositional atoms $\{q_1, \ldots, q_n\}$ are exactly those that appear in $I(x) = \Phi_x$.

   The symbols $\{q_1, \ldots, q_n\}$ are distinct from the atomic symbols of S.

We want to view (S, R, I) as a more general network than (S, R) and would like to define a sensible notion of complete extensions for it. We begin by explaining our strategy:

We are given an instantiated system of the form (S, R, I), where I is a function associating entities of the form **e** from some space **E**. We assume these entities can interact among themselves and that as a result of the interaction we can get truth values in, say, valued in the unit interval $[0, 1]$.

Strategy 1: Regard the attack relation R as stimulating interaction among the instantiated entities **e** and use the interaction to obtain values in [0,1] for the nodes of S.[6]

Strategy 2: Use the relationships among the entities **E** to change (S, R, I) into a new (S^*, R^*, I^*) and proceed with Strategy 1 for the new system.

There are several options and to explain the differences we need to be very precise.

Definition 2.9

1. *Let S be a set of arguments. Let Q_S be a corresponding set of atomic propositions of the classical propositional logic of the form*

$$Q_S = \{q_x | x \in S\}$$

where q_x are distinct symbols, i.e., $x \neq y \Rightarrow q_x \neq q_y$ and $S \cap Q_X = \emptyset$. Note that when there is no possibility of confusion, we abuse notation and write "x" in place of "q_x".

2. *Let I_{id} be the function $I_{id}(x) = q_x$, or, by abuse of notation, $I(x) = x$.*

Our objective is to define the notion of complete extensions for $B - AIAF$s of the form (S, R, I). There are several main views we can take:

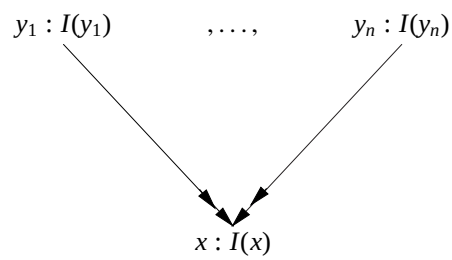
View 1: The $(E < I)$ view. First take a traditional complete extension E of (S, R) and then instantiate the element of E. We get a set T_E of classical propositional formulas. We need this set to be logically consistent.

This is what we did with the system of Figures 9a and 9b.

We needed the remedies discussed in Section 1, which essentially abandoned this view in favour of the next view $(I < E)$ which says, first instantiate then take extensions. We shall discuss this view next.

View 2: The $(I < E)$ view. We follow Strategy 1 here and keep (S, R, I) intact as is. We view R as stimulating interactions among the instantiating entities and try to extract numbers from the interaction. We use the Equational approach to do that. This view instantiates first and then takes extensions. Obviously we need to make use of the instantiation when we consider how to define the extensions. Let us explain our options using our experience with the examples we have from Section 1.

[6] There are more details in the beginning of appendix E1. See Example E.1 and the paragraph preceding it.



$y_1, \ldots, y_n$ are all the attackers of x.

Figure 19:

Consider (S, R, I) and consider the situation in Figure 19. The nodes $\{x, y_1, \ldots, y_n\}$ are all from (S, R), where $y_1, \ldots, y_n$ are all the attackers of x. The figure shows also the instantiation $I(x)$ and $I(y_1), \ldots, I(y_n)$.

We need to give meaning to the statement ($\sharp$1)

$$I(y_1), \ldots, I(y_n) \text{ attack } I(x) \tag{$\sharp$1}$$

The basic meaning of the atomic statement

$$y_1, \ldots, y_n \text{ attack } x \tag{$\sharp$1 atomic}$$

is that

$$x = 1 \text{ (“in”) iff all } y_1, \ldots, y_n \text{ are equal } 0 = \text{(“out”)}.$$

We consider several options for understanding such attacks.

View 2 - option 1. The equational approach. We follow Strategy 2 here and rewrite (S, R) in a more convenient form, using the structure of the instantiated entities. In this case we simplify/eliminate negation, by moving to a two state networks. This view was proposed in my paper on the equational approach to contrary to duty obligations [1]. The first section of that paper is general theory and regards Figure 19 as generating the Boolean (or real valued $[0, 1]$) equation

$$Eq(x) : I(x) \leftrightarrow [\bigwedge_{i=1}^{n} \neg I(y_i)].$$

The system of equations $\{Eq(x) | x \in S\}$ may or may not have a solution in the space $\{0, \frac{1}{2}, 1\}$.

Any such solution **f** is considered a complete extension for (S, R, I).

Note that the solution **f** gives values to the atoms of the logic, i.e. it is a 3-valued model of propositional logic. These values can be propagated to the formulas of the form $I(x), x \in S$, and the value of $I(x)$ under **f** can be viewed as the argumentation value of x under the complete extension **f**.

For example, suppose our network contains Figure 19 and that we have $I(y_i) = \neg q$ and $I(x) = q$. Then the equation for node x is $q = \neg q$ and therefore any solution $\mathbf{f}$ to the overall system of equations (for the network in which Figure 19 resides) will have to give the three valued assignment $q = \frac{1}{2}$ to q. The value of $\neg q$ is then $(1 - \frac{1}{2}) = \frac{1}{2}$ and hence the value of $I(x) = \frac{1}{2}$.

Therefore the node x is considered undecided in (S, R, I) under the solution (complete extension) $\mathbf{f}$.

This is a sweeping general option. Let us see what it does to Figure 10.

We have the following equations:

- $A(J) = 1$
- $A(J) \vee A(M) = \neg A(J)$
- $A(M) = \neg(A(J) \vee A(M))$

There is no solution to these equations. We are not surprised. The equational approach agrees with and generalises the traditional approach and so the inconsistencies and problems remain.

We need the remedies hinted at in Section 1.

View 2 - option 2. the two state equational approach. We follow Strategy 2 here and rewrite (S, R) in a more convenient form, using the structure of the instantiated entities. In this case we simplify/eliminate negation, by moving to a two state networks.

This is the approach we adopted in Example 1.4, executed within the equational framework. To show what it does, we modify first Figure 10 into a two-state associated figure and then use equations. We get Figure 20.

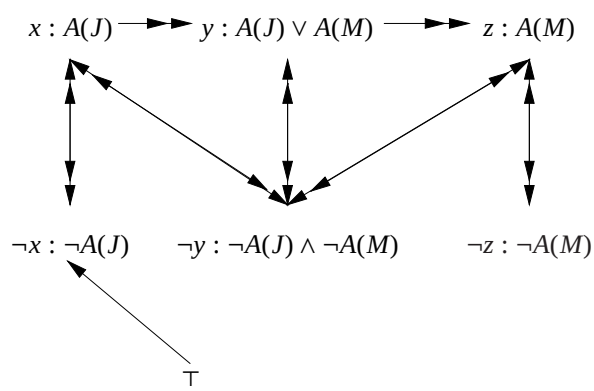


Figure 20:

The equations are

- $\top = 1$
- $\neg A(J) = \bot \wedge \neg A(J)$

- $A(J) = \neg\neg A(J) \wedge \neg(\neg A(J) \wedge \neg A(M))$.
- $\neg A(J) \wedge \neg A(M) = \neg(A(J) \vee A(M))$
- $A(J) \vee A(M) = \neg A(J) \wedge \neg(\neg A(J) \wedge \neg A(M))$
- $A(M) = \neg(A(J) \vee A(M)) \wedge \neg\neg A(M) \wedge \neg(\neg A() \wedge \neg A(M)$
- $\neg A(M = \neg A(M)$.

There is one solution $A(J) = A(M) = 0$.

We need to explain how and why, for example, $\neg y : \neg A(J) \wedge \neg A(M)$ is attacking $x : A(J)$ and $z : A(M)$ and vice versae. We added these attacks. The reason for this addition should be syntactical, not because we use logic. Formal definitions need to be given for the syntactical pattern matching.

To explain how this is done, let us do again the analysis of Figure 10. This time we write all wffs in disjunctive normal form. We get Figure 21.

$$\begin{tikzcd}
x : (A(J) \wedge A(M)) \vee (A(J) \wedge \neg A(M)) \arrow[d] \\
y : (A(J) \wedge A(M)) \vee (A(J) \wedge \neg A(M)) \vee (\neg A(J) \wedge A(M)) \arrow[d] \\
z : (A(J) \wedge A(M)) \vee (\neg A(J) \wedge A(M))
\end{tikzcd}$$

Figure 21:

Rewriting as a two state network we get Figure 22

$$\begin{tikzcd}
 & \top \arrow[dl] & \\
x : (A(J) \wedge A(M)) \vee (A(J) \wedge \neg A(M)) \arrow[d] \arrow[r, leftrightarrow] & \neg x : (\neg A(J) \wedge A(M)) \vee (\neg A(K) \wedge \neg A(M)) & \\
y : (A(J) \wedge A(M)) \vee (A(J) \wedge \neg A(M)) \vee (\neg A(J) \wedge A(M)) \arrow[r, leftrightarrow] & \neg y : \neg A(K) \wedge \neg A(M) & \\
z : (A(J) \wedge A(M)) \vee (\neg A(J) \wedge A(M)) \arrow[r, leftrightarrow] & \neg z : (A(J) \wedge \neg A(M)) \vee (\neg A(J) \wedge \neg AM)) &
\end{tikzcd}$$

Figure 22:

We can now add attacks to Figure 22 using pattern recognition.

- $e_1 : \bigvee \alpha_i$ attacks $e_2 : \bigvee \beta_j$ if they don't have any γ in common, i.e. it is not the case that for some $i, j, \gamma = \alpha_i = \beta_j$.

We get Figure 23. Note that $\top$ is an exception to this rule.

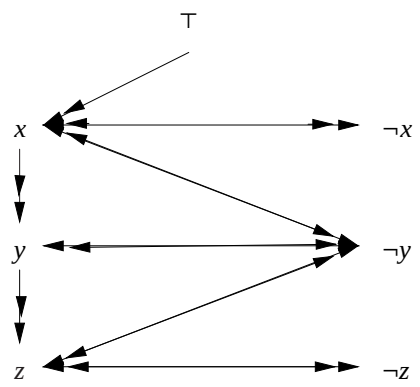


Figure 23:

The perceptive reader might complain that we are nevertheless using logic, i.e. we are using resolution theorem proving. The answer is that we are not. Resolution is a discipline of sequencing various pattern matchings. Just comparing two disjunctive normal forms does not, in itself, make a resolution theorem prover.

So the steps in View 2, option 2 are as follows:

1. Start with (S, R, I).
2. Take the associated (S^*, R^*, I^*) explained above (formal definitions to come later).
3. Use the equational approach to find the extensions.

View 3. Direct computation approach. In this approach we develop the concept of semantics and extensions directly on the instantiated network by translating it (with the help of additional arguments) into traditional Dung networks or into a modified/generalised such network. This requires as a by product the translation of the entities **E** into abstract argumentation, either directly, or indirectly.

It also means that we are turning the instantiation problem into a fibring problem in the sense of [3].

2.2 Concrete classical propositional instantiations

We are going to give a progression of challenges for instantiations from the classical propositional calculus. Many of these instantiations have been dealt with in the Appendices. Here we summarise the big picture.

Challenge 1: Instantiation with $\top$

This has been defined and given semantics in Appendix B.

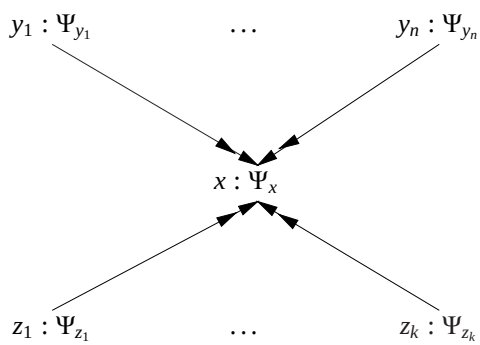


Figure 24:

Challenge 2: Instantiate with conjunctions of atomic propositions

Given a traditional network (S, R) we instantiate with a function I defined on S, giving for each $x \in S$ a conjunction Ψ_x of atomic propositions of the form $\Psi_x = \bigwedge_{i=1}^{n(x)} q_i^x$. We also write Ψ_x as a set $\{q_1^x, \ldots, q_{n(x)}^x\}$. The basic geometrical position we get is as in Figure 24, which should be compared with Figures 19 and 50.

We offer semantics for the instantiated system (S, R, I) by using instead of Ψ_z the $\mathbb{BF}_z$ of Figure 51 and Figure 59 of Appendix C2.

We then implement the instantiation of the $\mathbb{BF}_z$ (i.e. $I'(z) = \mathbb{BF}_z, z \in S$) as proposed in Definition C.10 of Appendix C3.

We adopt the semantics of option (iv) of Appendix B as discussed there.

Challenge 3: Instantiating with disjunctions

When we instantiate with wffs containing disjunctions, we get attacks of the form

$$(A \vee B) \twoheadrightarrow (C \vee D)$$

a disjunction attacking another disjunction. The form $A \vee B \twoheadrightarrow z$ is equivalent to $A \twoheadrightarrow z$ and $B \twoheadrightarrow z$. So we need to deal with the case of the form $x \twoheadrightarrow C \vee D$ and try and eliminate or implement the disjuction. The semantic consition $x \twoheadrightarrow C \vee D$ from the equational point of view is

$$C \vee D \equiv \neg x$$

or

$$\neg C \wedge \neg D \equiv x$$

or

$$\neg C \wedge \neg D \equiv \neg(\neg x).$$

This means that $x \twoheadrightarrow C \vee D$ is equivalent to $\neg x \twoheadrightarrow \neg C \wedge \neg D$.

We already know how to attack conjunctions from Challenge 2. So we need to deal with negation. Once we do that we will be able to deal with full Boolean instantiation.

Challenge 4: Instantiating with negated formulas

Let us start with (S, R). We move to (S^*, R^*) as defined in Definition 1.2 and for and for which Lemma 1.3 holds. (S^*, R^*) is a two state network. For every $x \in S^*$, there is a node $\neg x \in S^*$, with $x \twoheadrightarrow \neg x$ and $\neg x \twoheadrightarrow x$, and where $\neg\neg x$ is x. Thus any instantiation $I(x) = \Phi$ for $x \in S$ becomes the double instantiation I^* on S^* where $I^*(x) = \Phi$ and $I^*(\neg x) = \neg\Phi$.

We now have a system (S^*, R^*, I^*) of instantiated Boolean net. We use Appendices C2 and C3, Example C.8 and Definition C.10 to replace the wff $I(x)$ for $x \in S^*$ with a Boolean formulation $\mathbb{BF}_x$ and then instantiate (S^*, R^*) with $x \mapsto \mathbb{BF}_x$. The resulting system is a $\top$-net as discussed in Appendix B and we can calculate option (iv) extensions for it.

Remark 2.10 *Let us summarise how to get and what it means to be extensions when we instantiate any network (S, R) with formulas from classical propositional logic, using any of the above challenges (depending on the wffs used in the instantiations).*

Let the instantiating function be $I : S \mapsto$ wffs, where wffs belong to a language with atoms Q. We replace I by I^* giving each x a $\mathbb{BF}_x$ which basically does the same job (says the same) as $I(x) = \Phi_x$. These $\mathbb{BF}$s were done in Appendix C. $\mathbb{BF}_x$ contains the atoms appearing in Φ_x (atoms from Q) as well as many auxiliary atoms.

(S, R) is replaced by (S^*, R^*) being a network of instantiated $\mathbb{BF}$s containing the propositional atoms of all the Φs $\{\Phi_x | x \in S\}$ (i.e. all the atoms of Q), as well as many auxiliary atoms. We use appendix B to find extensions for (S^*, R^*). These are functions λ, giving values in $\{0, 1, \frac{1}{2}\}$ to all atoms in (S^*, R^*) and thus giving values to all the atoms Q of the propositional language. Once we have values for the atoms we have a 3-valued propositional model and we get values for all the wffs $\Phi_x, x \in S$. Define an extension λ^* on (S, R) by $\lambda^*(x) = \lambda(\Phi_x)$. We need to show that this is an extension in the sense of Definition 2.3. This follows from the way we set up the entire process. It does require proof but I will not do it now.

We thus got a 3-valued model for the language Q and an extension λ^* out of the instantiation I for (S, R).

So given a Boolean instantiation (S, R, I) what does it mean to have an extension for it?

It means a 3-valued model λ for the Boolean instantiation language such that $\lambda^*(x) = \lambda(\Phi_x)$ is an extension of (S, R).

2.3 Instantiating with monadic wffs and modal S5 wffs

Given (S, R) we want to instantiate with wffs Φ of monadic logic. At the first instance we assume monadic logic with $\{P_1, \ldots, P_n\}$ and we assume that Φ has no free variables. We make use of Appendix A.

Let I be an instantiation function giving each $x \in S$ a closed formula $I(x) = \Phi_x$ of the monadic predicate logic based on $\{P_1, \ldots, P_n\}$.

We seek extensions for the instantiated system (S, R, I).

We use Lemma A.6, which says that every wff Φ without free variables is a Boolean combination of "atomic" wffs of the form $q_\varepsilon = \exists x \alpha_\varepsilon(x)$, where $\alpha_\varepsilon(x)$ has the form $\bigwedge_{i=1}^n P_i^{e_i}(x)$ where $\varepsilon = (e_1, \ldots, e_n) \in 2^n$.

We can thus pretend we are dealing with classical propositional logic with 2^n atomic formulas of the form $q_\varepsilon, \varepsilon \in 2^n$.

Associate with each closed formula $\Phi = \Phi(\exists x \alpha_\varepsilon(x))$ the propositional formula

$$\Phi^* = \Phi(\exists x \alpha_\varepsilon(x)/q_\varepsilon).$$

We now instantiate (S, R) with the formulas Φ^* instead of Φ. I.e. we look at (S, R, I^*) where $I^*(x) = \Phi^*_x$. Get an extension λ on the atoms $\{q_\varepsilon\}$. From it get an extension λ^* on S. Also since we have λ values for $\{q_\varepsilon\}$ we get values for all the predicate "atomic" formulas of the form $\exists x \alpha_\varepsilon(x), \varepsilon \in 2^n$ and thus get a 3-valued predicate model "instantiating" the net (S, R, I).

The case where the formula has free variables , we regard them as constants and proceed using Remark A.7. The case of modal S5 is treated similarly in view of Remark A.8.

2.4 Beyond predicate instantiation

Our methodological approach, so far as discussed in Section 1.2, was to use theoretical considerations in expanding our argumentation theory to predicate instantiation. We simply substituted formulas of monadic predicate logic into an abstract argumentation network (S, R) and asked how can we deal with it on theoretical grounds.

We now want to check what kind of predicate networks are required by day to day practical applications. We use Example 2.11 as a starting point. This type of examples arise in Talmudic logic, see [22].

Example 2.11 *At home I have several sinks and 3 toilets. If a sink is blocked, I can handle it myself. If a toilet is blocked, it is reasonable that I call a plumber. Two opposing principles come to play here. A plumber costs money (about US$ 100 just to visit in addition to any other charges depending on the job). So it makes sense for me to try and do the job myself if I can. So with a simple case like a blocked sink I can do it myself, but with a blocked toilet I had better call a plumber.*

Let us write these rules:

1. $B(s) \to \neg\exists x P(x, s)$
2. $B(t) \to \exists P(x, t)$

where $B(z)$ reads z is blocked and $P(x, z)$ reads x is called to repair z. The $\to$ is ordinary implication. Figure 25 shows these rules in argumentation form.

Now let us check what happens if both the sink and the toilet are blocked. Common sense dictates, that since I have to call a plumber to do the toilet anyway, I may as well ask him to do the sink. I am paying the $ 100 for the visit anyway. In comparison, the strict logical solution to the problem is that I do the sink and the plumber does the toilet! (This is applying principles (1) and (2).)

Furthermore, if two of my toilets are blocked, the above rules (1) and (2) allow me to called different plumbers, one for each toilet rather than call the same plumber to do all jobs.

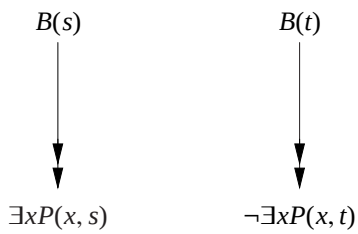


Figure 25:

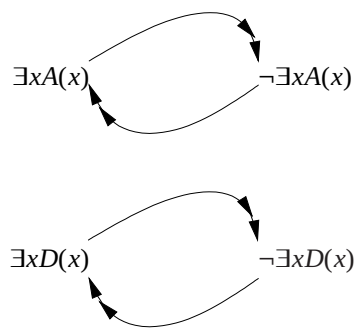


Figure 26:

So what we need formally in the predicate network is that if $\exists x A(x)$ is "in" because of $x = a$, then every other $\exists x D(x)$ which is "in" must be instantiated by the same $x = a$!

So let us forget about plumbers and just look at the network of Figure 26

We require that if one of $\exists x A(x)$ or $\exists x D(x)$ is "in", the other is also "in" and because of the same element a. (This reflects our analysis that if you have many sinks and toilets and you need to call a plumber for one of the toilets then you ask this same plumber to do everything.)

General problem. We are given a predicate network (S, R, I) and a subset $S_0 \subseteq S$ such that for all $s \in S_0$, $I(s) = \exists x D_s(x)$ (with D_s a unary predicate depending on s). We want to implement the additional constraint

$\mathbb{C}$: If for some $s \in S_0$, $I(s)$ is "in" and for some a, $D_s(a) = \top$ then for all $s \in S_0$ we have that $I(s) =$ "in" and furthermore for that same a, $D_s(a) = \top$.

Implementing that part of $\mathbb{C}$ which says

if $I(s_1)$ is "in" then also $I(s_2)$ is "in"

is not simple. We might think that we can implement that by letting $I(s_1)$ attack $\neg I(s_2)$. But this is correct only because $I(s_1)$ and $I(s_2)$ are instantiated as existential statements. So we cannot write the attack in the geometry of (S, R). It must occur/activate after the instantiation. How do we do that?

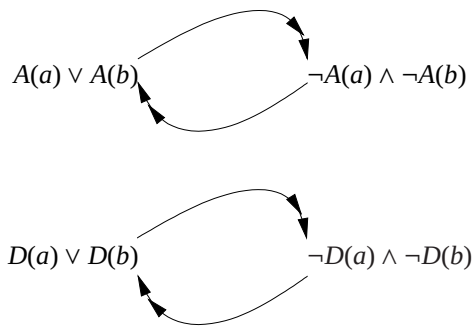


Figure 27:

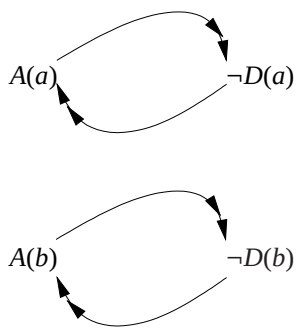


Figure 28:

To get a better idea, let us assume that our universe of elements has only two elemnts $\{a, b\}$. This would allow us to rewrite any $\exists x E(x)$ as $E(a) \vee E(b)$, and so Figure 26 becomes Figure 27.

We know how to handle full propositional instantiations from Section 2.2. In this case we have the additional constraints:

1. $A(a) = 1$ iff $D(a) = 1$
2. $A(b) = 1$ iff $D(b) = 1$
   In general we want a condition like
3. $\forall x(A(x) = 1$ iff $D(x) = 1)$.

This means we add to Figure 27 also the attacks of Figure 28. Even if we could implement this using various geometrical attacks, we still have two problems:

P1 It all depends on the instantiation.

P2 It applies only to points in S_0.

It looks like we need a new fresh breakthrough point of view, otherwise we will have to restrict the acceptable complete extensions by conditions written in the metalevel and not by the object level properties of the network.

$$E_1(x) \longrightarrow\!\!\!\!\rightarrow E_2(x)$$

$\alpha(x)$ is a constraint on x. Only certain values of x can attack from E_1 to E_2. Those x which satisfy $\alpha(x)$.

Figure 29:

Our problems are not over yet. We saw we needed the condition

$$\forall x(A(x) = 1 \text{ iff } D(x) = 1).$$

This means in attack terms

$$\forall x(A(x) \twoheadleftarrow\!\!\!\twoheadrightarrow \neg D(x)).$$

But how do we implement the general principle like a universal quantification of the form $\forall x(A(x) \twoheadleftarrow\!\!\!\twoheadrightarrow \neg D(x)$?

We will have to add free variable attacks to networks as in Figure 29.

This is a new twist in our conceptual analysis of what is happening here.

In Section 2.3 we treated free variables in an argumentation networks as constants. Now we see we need to treat them as parameters, and the attack arrows are annotated by these parameters.

This is a new game to be properly analysed.

References

[1] D. Gabbay. The Equational approach to contrary to duty obligations, in *Foundations of Information and Knowledge Systems*, C. Beierle and C. Meghini, eds. Lecture Notes in Computer Science Volume 8367, 2014, pp 1-61.

[2] G. Brewka and S. Woltran. Abstract dialectical Frameworks, in *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2010)*, pp 102-111.

[3] D. Gabbay. Fibring Argumentation Frames. *Studia Logica*, 93(2-3): 231–295,2009.

[4] D. Gabbay. An Equational Approach to Argumentation Networks, Feb 2011, 104 pp . Paper split into two papers: First part published in *Argumentation and Computation*, 2012, vol 3 issues (2-3), pp 87-142.

[5] D. Gabbay and O. Rodrigues. An equational approach to the merging of argumentation networks. *J Logic Computation* first published online November 20, 2013. doi:10.1093/logcom/ext060(25 pages)

[6] D. Gabbay. The Equational approach to Logic Programs, in *LNCS 7265 Festschrift for Vladimir Lifschitz*, Esra Erdem, Joohyung Lee,Yuliya Lierler and David Pearce, eds., pp 279–295, 2012

[7] D. Gabbay. *Meta-Logical Investigations in Argumentation networks*. Research Monograph College publications 2013, 770 pp

[8] Nikos Gorogiannis, Anthony Hunter. Instantiating abstract argumentation with classical logic arguments: Postulates and properties, *Artificial Intelligence* 175 (2011) 1479–1497

[9] Francesca Toni and Marek Sergot. Argumentation and Answer Set Programming. In *Gelfond Festschrift, LNAI 6565*, M. Balduccini and T.C. Son, eds. pp. 164–180, 2011. Qc Springer-Verlag Berlin Heidelberg 2011

[10] Hannes Strass. Approximating operator sand semantics for abstract dialectical frameworks *Artificial Intelligence*205(2013)39–70

[11] M Caminada and D Gabbay. A logical account of formal argumentation *Studia Logica*, 93(2-3): 109–145, 2009.

[12] Yining Wu, Martin Caminada, and Dov M. Gabbay. Complete Extensions in Argumentation Coincide with 3-Valued Stable Models in Logic Programming *Studia Logica*, December 2009, Volume 93, Issue 2-3, pp 383-403

[13] Sanjay Modgil. Revisiting Abstract Argumentation Frameworks Theory and Applications of Formal Argumentation *Lecture Notes in Computer Science* Volume 8306, 2014, pp 1-15

[14] Henry Prakken. An abstract framework for argumentation with structured arguments, *Argument and Computation*, 1:2, 93-124, 2010. DOI: 10.1080/19462160903564592

[15] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and *n*-Person Games, *Artificial Intelligence*, 77, 321–357, 1995.

[16] Martin Caminada and Leila Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence* 171, 5–6, pp. 286–310, 2007.

[17] Caminada 2006 (from quoted material)

[18] Søren Holbech Nielsen and Simon Parsons. A Generalization of Dung's Abstract Framework for Argumentation: Arguing with Sets of Attacking Arguments in *Argumentation in Multi-Agent Systems*, Lecture Notes in Computer Science Volume 4766, 2007, pp 54-73.

[19] Stephen Cole Kleene *Introduction to Metamathematics*, 1952.

[20] P. Baroni, M. Giacomin, G. Guida. SCC-recursiveness: a general schema for argumentation semantics. *Artificial Intelligence*, 168(1–2), 162–210, 2005.

[21] Dov Gabbay. Logical foundations for bipolar argumentation networks, to appear in special issue of the *Journal of Logic and Computation* in honour of Arnon Avron. *Journal of Logic and Computation* 2013; doi: 10.1093/logcom/ext027

[22] M. Abraham, I. Belfer, D. Gabbay and U. Schild. Identity merging and identity revision in Talmudic logic. To appear in *Annals of Mathematics and Artificial Intelligence*, special issue on Belief Change and Argumentation Theory, edited by Juergen Dix, Sven Ove Hansson, Gabriele Kern-Isberner and Guillermo Simari.

[23] Leila Amgoud and Caroline Devred. Argumentation Frameworks as Constraint Satisfaction Problems, in S. Benferhat and J. Grant (Eds.): *SUM 2011*, LNAI 6929, pp. 110Ð122, 2011.

[24] Stefano Bistarelli, Daniele Pirolandi and Francesco Santini. Solving Weighted Argumentation Frameworks with Soft Constraints. In *Recent Advances in Constraints*, Lecture Notes in Computer Science Volume 6384, 2011, pp 1-18

[25] Caroline Devred, Sylvie Doutre, Claire Lefévre and Pascal Nicolas. Dialectical Proofs for Constrained Argumentation, COMMA 2010.

[26] Dov Gabbay and Tjitze Rienstra. Intervention and observation in argumentation networks, to appear special issue of *Logica Universalis*.

[27] Tjitze Rienstra. *Argumentation in Flux*. Thesis, a revised version to appear with College Publications.

[28] Ofer Arieli, Christian Strasser. Dynamic derivations for sequent-based logical argumentation. *Proc. 5th International Conference on Computational Models of Argument (COMMA'14), Frontiers in Artificial Intelligence and Applications*, IOS Press, 2014. To Appear

[29] Christian Strasser and Ofer Arieli. Sequent-based argumentation for normative reasoning. *Proc. 12th International Conference on Deontic Logic and Normative Systems (DEON'14)*, LNAI 8554, pages 224–240, Springer, 2014.

[30] Philippe Besnard and Anthony Hunter. *Elements of Argumentation*, MIT Press, 2008.

[31] Sanjay Modgil and Henry Prakken. The ASPIC+ framework for structured argumentation: a tutorial. *Argument and Computation*, 5:1, 31–62, 2014. Special Issue: Tutorials on Structured Argumentation, DOI:10.1080/19462166.2013.869766

[32] D Gabbay. Compromise Update and Revision A position paper. In *Dynamic Worlds*, B. Fronhoffer and R. Pareschi, eds. pp. 111–148. Kluwer, 1999.

[33] D Gabbay, O. Rodrigues and A. Russo. *Revision, Acceptability and Context: Theoretical and Algorithmic Aspects* (Cognitive Technologies) Springer 2010.

[34] Dov Gabbay. *Labelled Deductive Systems; Principles and Applications. Vol 1: Basic Principles*, Oxford University Press, 1996.

[35] G. Brewka, S. Ellmauthaler, H. Strass, J. P. Wallner and S. Woltran. Abstract dialectical frameworks revisited. In *Proc. IJCAI 2013*, pp. 803–809, AAAI Press, 2013.

[36] D. Gabbay and O. Rodrigues. The equational approach to abstract dialectical frameworks. To appear.

[37] D. Gabbay. Dung's argumentation is equivalent to classical propositional logic with the Peirce-Quine dagger. *Logical Universalis*, 5-, 255–318, 2011.

Appendices

A Classical monadic predicate logic

The purpose of this appendix is to show point 3 of Section 1.3, namely that every formula of monadic predicate logic is equivalent to a formulas in a (argumentation friendly) syntactic form, Lemma A.6 below.

Definition A.1

1. *The language of $\mathbb{L}_n$, of the classical monadic predicate logic without equality has n unary predicates $P_1, \ldots, P_n$, variables the classical connectives $\{\neg, \wedge, \vee, \rightarrow\}$ and the quantifiers $\{\forall, \exists\}$ with their usual meaning.*

2. *A model for $\mathbb{L}_n$ has the form $\mathbb{M} = (D, D_1, \ldots, D_n)$, where D is a non-empty domain and $D_i \subseteq D$ is the extension of the predicate P_i.*

We define satisfaction in a model in the traditional way, using the traditional abuse of notation as follows:

- $\mathbb{M} \vDash P_i(d)$ *iff* $d \in D_i$
- $\mathbb{M} \vDash \neg A$ *iff* $\mathbb{M} \nvDash A$
- $\mathbb{M} \vDash A \wedge B$ *iff* $\mathbb{M} \vDash A$ *and* $\mathbb{M} \vDash B$
- $\mathbb{M} \vDash A \vee B$ *iff* $\mathbb{M} \vDash A$ *or* $\mathbb{M} \vDash B$
- $\mathbb{M} \vDash A \rightarrow B$ *iff* $\mathbb{M} \vDash A$ *implies* $\mathbb{M} \vDash B$
- $\mathbb{M} \vDash \exists x A(x)$ *iff for some* $d \in D, \mathbb{M} \vDash A(d)$
- $\mathbb{M} \vDash \forall x A(x)$ *iff for all* $d \in D, \mathbb{M} \vDash A(d)$.

Lemma A.2 *Let $\mathbb{M}$ be a model. Then there exists a model $\mathbb{M}^*$ with at most 2^n elements which is equivalent to $\mathbb{M}$.*

Proof Define $\approx$ on D by

- $x \approx y$ iff for all $i, P_i(x) \leftrightarrow P_i(y)$.

Then $\approx$ is an equivalence relation. We show that

$$\begin{array}{l}\text{For every } A(x_1, \ldots, x_m), \text{ if } x_j \approx y_j, \text{ for } j = 1, \ldots, m \\ \text{then } \mathbb{M} \vDash A(x_1, \ldots, x_m) \leftrightarrow A(y_1, \ldots, y_m).\end{array} \qquad (*)$$

Proof of (*). By induction on the syntactical structure of A.

Let D^* be the set of $\approx$ equivalence classes of elements of D. Similarly let D_i^* be the set of equivalence classes of elements of D_i.

Let x^* be the equivalence class of x.

Let $\mathbb{M}^* = (D^*, D_1^*, \ldots, D_n^*)$.

We prove the following.

For any $x_j \in D, \mathbb{M} \vDash A(x_j)$ iff $\mathbb{M}^* \vDash A(x_j^*)$.

Proof is by induction on A. ■

$\mathbb{M}^*$ has at most 2^n elements because there are at most 2^n possibilities for $(\pm P_1(x), \ldots, \pm P_m(x))$ for any single x.

Definition A.3 *Two models $\mathbb{M}$ and $\mathbb{N}$ are said to be equivalent if $\mathbb{M}^* = \mathbb{N}^*$.*

Definition A.4

1. *Let ε denote a vector of length n of elements $e_i \in \{0, 1\}$. Thus*

$$\varepsilon = (e_1, \ldots, e_n) \in 2^n.$$

   ε is called a type.

2. *For any $\varepsilon \in 2^n$, let $\alpha_\varepsilon(x)$ be the wff*

$$\alpha_\varepsilon(x) = \bigwedge_{i=1}^{n} P_i^{e_i}(x)$$

   where $P^0(x) = \neg P(x)$ and $P^1(x) = P(x)$.

   This means that x is of type ε.

3. *Let Γ be a non-empty set of types, i.e. ε-vectors, $\Gamma \subseteq 2^n$.*

Lemma A.5 *Let $\mathbb{M}$ be a model of $\{P_1, \ldots, P_n\}$. Then $\mathbb{M}$ is characterised by a formula of the form*

$$\Phi_{\mathbb{M}} = \bigwedge_{\varepsilon \in \Gamma_{\mathbb{M}}} \exists x \alpha_\varepsilon(x) \wedge \bigwedge_{\varepsilon \notin \Gamma_{\mathbb{M}}} \neg \exists x \alpha_\varepsilon(x).$$

In words, $\mathbb{M}$ is characterised by the types it realises where $\Gamma_{\mathbb{M}}$ is defined as

$$\{\varepsilon \mid \textit{for some } d \in \mathbb{M}, \mathbb{M} \vDash \alpha_\varepsilon(d)\}.$$

Proof

1. Clearly $\mathbb{M} \vDash \Phi_{\mathbb{M}}$

2. Let $\mathbb{N}$ be a model such that $\mathbb{N} \vDash \Phi_{\mathbb{M}}$. Since $\Phi_{\mathbb{M}}$ says exactly for every ε whether $\exists x \alpha_\varepsilon(x)$ holds or not, we get that $\mathbb{N}$ satisfies the same types as $\mathbb{M}$. Therefore we have $\mathbb{M}^* = \mathbb{N}^*$.

■

Lemma A.6 *Let Φ be any formula of the language of $\{P_1, \ldots, P_n\}$ without free variables. Then Φ is equivalent to a wff of the form*

$$\Phi = \bigvee_j \Phi_{\Gamma_j}$$

where $\Gamma_j \subseteq 2^n$ and Φ_{Γ_j} is defined as follows

$$\Phi_{\Gamma_j} = \bigwedge_{\varepsilon \in \Gamma_j} \exists x \alpha_\varepsilon(x) \wedge \bigwedge_{\varepsilon \notin \Gamma_j} \neg \exists x \alpha_\varepsilon(x).$$

Proof By Lemma A.2 it is sufficient to consider models of Φ of less than 2^n elements. Let $\mathbb{M}_1, \ldots, \mathbb{M}_k$ be all the models of Φ. then the wff Φ is equivalent to $\bigvee_{j=1}^k \Phi_{\mathbb{M}_j}$ which has the same models as Φ. ■

Remark A.7 *In case the language contains propositional constants $q_1, \ldots, q_m$ as well as monadic predicates $P_1, \ldots, P_m$ then a model $\mathbb{M}$ for this language is characterised by a conjunction of the form*

$$\Phi_{\mathbb{M}} = \bigwedge_{\varepsilon \in \Gamma_{\mathbb{M}}} \exists x \alpha_\varepsilon(x) \wedge \bigwedge_{\varepsilon \notin \Gamma_{\mathbb{M}}} \neg \exists x \alpha_\varepsilon(x) \wedge \beta_{\mathbb{M}}$$

where $\Gamma_{\mathbb{M}}$ is as in Lemma A.5 and

$$\beta_{\mathbb{M}} = \bigwedge_{j=1}^m q_j^{e_j}$$

where $\eta_{\mathbb{M}} = (e_1, \ldots, e_m)$ is the vector in 2^m of the atoms q_j or their negations which hold in the model $\mathbb{M}$.

Therefore any formula Φ of the language with the constants $\{q_j\}$ is equivalent to a disjunction

$$\bigvee_j \Phi_{\Gamma_j} \wedge \beta_j$$

where Φ_{Γ_j} are as in Lemma A.6 and β_j is a formula

$$\beta_j = \bigwedge_r q_r^{e_r^j}$$

as discussed above.

If Φ is a formula with free variables $x_1, \ldots, x_k$, we regard the free variables as constants and regard $P_i(x_j)$ $i = 1, \ldots, n$, $j = 1, \ldots, k$ as propositional constants $q_{i,j} = P_i(x_j)$.

We thus can construct an equivalent formula as done above for monadic logic with constants $q_1, \ldots, q_m$, $m = k \times n$.

Figure 30:

Remark A.8 *The connection between monadic predicate logic with $\{P_1, \ldots, P_n\}$ and the modal logic S5 based on the atomic propositions $\{P_1, \ldots, P_n\}$ is well known. For an S5 Kripke model with a set of possible worlds D, let $d \vDash P_i$ be interpreted as $P_i(d)$, in the monadic theory based on D.*

Following this correspondence, let β_ε, for $\varepsilon \in 2^n$ be $\Diamond(\bigwedge_i P_i^{e_i})$ nd let β'_ε be $\bigwedge_j P^{e_i}$ where $\varepsilon = (e_1, \ldots, e_n)$. We get therefore

(*) Every wff of modal logic S5 with atoms $\{P_1, \ldots, P_n\}$ is equivalent to a wff of the form

$$\bigvee_j (\beta'_{\varepsilon_j} \wedge \bigwedge_{\varepsilon \in \Gamma_j} \beta_\varepsilon \wedge \bigwedge_{\varepsilon \notin \Gamma_j} \neg \beta_\varepsilon)$$

for some $\Gamma_1, \ldots, \Gamma_k \subseteq 2^n$ and for some β'_{ε_j} in Γ_i respectively.

(*) holds because an S5 model is a set of worlds containning the actual world. Each world is characterised by a conjuction of the form $\bigwedge_i P_i^{e_i}$, where $\varepsilon = (e_1, \ldots, e_n)$. We can identify a world with ε. A model is characterised by the set of worlds Γ it contains in conjuction with the set of worlds it does not contain (the complement of Γ). Thus a model is a conjuction $\beta'_{\varepsilon_i} \wedge \bigwedge_{\varepsilon \in \Gamma} \wedge \bigwedge_{\varepsilon \notin \Gamma_j} \neg \beta_\varepsilon)$. A formula is equivalent to several sets of worlds, i.e. a disjunction of several of the formulas characterising models.

The above observations will allow us to instantiate Dung argumentation frames with monadic predicate formulas or with S5 modal formulas.

B Instantiating with $\top$

This appendix discusses the subtleties of instantiating a single node in argumentation network with just $\top$. We shall see that new concepts of semantics and extensions are required for the proper handling of this seemingly innocent substitution.

To show that finding extensions for instantiated Boolean network is not that simple a task, let us start with a very simple example. Consider the network $(\{x, y\}, x \twoheadrightarrow y\}$. This has the extension $x =$ ‘in” and $y =$ “out”, i.e. $x \wedge \neg y$. Let us instantiate $y = \top$. We get the network of Figure 30. Let us refer to any network with atomic arguments S which also containns $\top$ as $\top$-net.

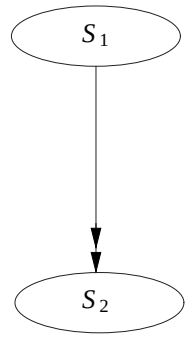


Figure 31:

So Figure 30 is a $\top$-net. We have a problem with this network. x is not attacked and hence x = "in". But x attacks $\top$ and $\top$ cannot be "out", it has to be "in". So what shall we do?

Further reflection shows that the problem is more serious than it seems at first sight. Traditional Dung extensions can be constructed using the geometrical directionality of the attack. Consider Figure 31.

We have $S = S_1 \cup S_2$ and attacks emanate from S_1 into S_2 and there are no attacks from S_2 into S_1. Thus we can find an appropriate inital extension E_1 for S_1 and propagate the attacks from E_1 into S_2 to complete the extension into S_2 and get a complete extension for $S_1 \cup S_2$. This is the directionality of the attack. We are always guaranteed an extension.

The situation we have now is that with $\top \in S_2$, the directionality no longer exists. Any attack from S_1 onto $\top \in S_2$, will force us to reconsider/change the extension E_1 of S_1. It is as if there are attacks from S_2 into S_1.

So what are our options in dealing with this? Let us go back to the network of Figure 30 and try and apply general principles to it giving us several options:

Option (i): Truth intervention view. We can say this network has no extensions. This position is perfectly acceptable. It is legitimate and reasonable to take a traditional network (S, R), pick a $y \in S$ and demand an extension E with $y \in E$. This is enforcing truth on y. We may find that no such extension can be found. So letting $y = \top$ amounts to saying that we want only extensions containing y.[7]

Let us call this option (i), the $\top$-intervention approach, because we are intervening and forcing some nodes to be true = "in".

Option (ii): Counter attack view. We can say that any attacker z of $\top$ is immediately attacked back by $\top$, and so the above Figure 30 is actually Figure 32. The extension is $\top$ = "in" and x = "out".

[7]We mention here reference [24], where they define the notion of constraint argumentation networks. Given a formula Φ of propositional logic, and a network (S, R), we accept only those extensions λ such that T_λ of Section 1.3, $(T_\lambda = \{q|\lambda(q) = 1\} \cup \{\neg q|\lambda(q) = 0\})$, satisfies Φ. In our case the formula Φ is $y = \top$.

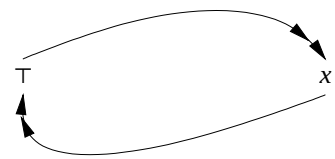


Figure 32:

The advantage of this view is that the usual traditional machinery for defining and finding complete extensions can be used, with the additional understanding that $\top$ is always "in" in any extension.

Option (iii): New concept of extension view. The third option is to give a new definition of abstract networks with truth constant $\top$ as follows:

1. (S, R) is a network with $\top$ if $\top \in S$ and $\neg\exists y(yR\top)$.
2. A Caminada legitimate $\top$-labelling for a network with $\top$ satisfies the following:
   (a) $\lambda(\top) = 1$
   (b) $\lambda(x) = 1$ if $\neg\exists y(yRx)$ and $\neg xR\top$.
   (c) $\lambda(x) = 0$ if $xR\top$
   (d) $\lambda(x) = 1$ if for all y such that yRx we have $\lambda(x) = 0$ and $\neg xR\top$
   (e) $\lambda(x) = 0$ if $xR\top$ or if for some y, yRx and $\lambda(y) = 1$.
   (f) Otherwise $\lambda(x) = \frac{1}{2}$.

There is the question of whether every $\top$-net have an $\top$-extension (i.e. a legitimate Caminada $\top$-labelling). The answer is yes, it does. Let (S, R) be a $\top$-net. Let $T = \{y \in S | \top Ry \vee yR\top\}$. We know all of these points should be out. Let $*$ be a point not in S and consider

$$S^* = S \cup \{*\} - \{\top\}$$
$$R^* = (R \restriction S - \{\top\}) \cup \{*\} \times T$$

In other words, we take $\top$ out and include $*$ which attacks all points in T. (S^*, R^*) is an ordinary network and has extensions. Let E be such an extension. Then $(E - \{*\}) \cup \{\top\}$ is an extension of the $\top$-net (S, R). So for example the network (S, R) of Figure 30 becomes the network with $S^* = \{*, x\}$ with $\{(*, x)\} = R^*$.

Option (iv): The non-toxic truth intervetion view. Let us adopt the option (i) view for a given $\top$-net (i.e. a network (S, R) with a node $y = \top \in S$) and seek only extensions containing $y = \top$. However, we shall adopt option (i) not alone on its own but adopt it in conjunction with another new principle, which we shall call the principle of *maximal non-toxic extensions* (*NTE principle* in short).

Examples B.1 and B.2 shall explain it.

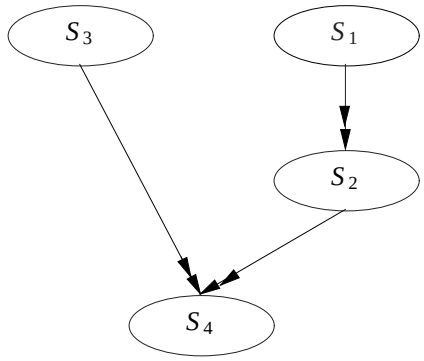


Figure 33:

Example B.1 *Consider Figure 33*

This figure describes a network with four sub-networks $S = S_1 \cup S_2 \cup S_3 \cup S_4$. *The networks* S_i *are pairwise disjoint and* S_2 *does not attack* S_1, S_4 *does not attack* S_2 *nor* S_3. S_2 *however, contains* $\top$. *Suppose for some specific extension* E_1 *of* S_1, *it is not possible to extend* E_1 *to an extension for* S_2. *This makes* S_2 *toxic for* E_1. S_2 *forces us to say that* S *has no extensions* E *with* $E \cap S_1 = E_1$.

However, if we ignore the toxic S_2, *we may get extensions* $E \supseteq E_1$ *for* $S_1 \cup S_3 \cup S_4$. *It makes sense to do that and present* E *as a maximal non-toxic extension for* S *containing* E_1 *for* S_1.

To motivate the logical sense of doing that, consider a perfectly nice network (S, R) *not containing the letter* x *nor the symbol* $\top$. *Add to* S *the letters* x *and* $\top$ *and augment* R *with* $x \twoheadrightarrow \top$ *(i.e. add the toxic Figure 30 to* (S, R)*). The resulting network has no exensions because of the toxic part. It does make sense, however, to say that if we ignore the toxic part, we can get extensions for* (S, R).

The perceptive reader might ask what if we add to (S, R) *a disjoint 3-cycle? Can we similarly ignore it? The anser is that we do not need to, because the three cycle does have the empty extension (all undecided) an so the traditional machinery works.*

Let us give an example from real life. Consider a couple going through a divorce in the UK. They want to settle the financial part amicably and so they go to an accountant. By UK law, if the accountant learns in the process of advising them of any tax evasion scheme he/she has to report it to the authorities. So the divorcing couple decide that some of their business is "toxic" and better not tell the accountant.

Similarly in a court case both prosecutor and defence lawyers may decide to drop some charges because it is too complicated/toxic for each side to address, each for their own respective reasons.

Example B.2 *Consider the* $\top$*-net of Figure 34. We want to examine the semantics for it according to option (iv), non-toxic truth intervention.*

This figure has the form of Figure 33 with $S_1 = \{x\}, S_2 = \{a, b\}, S_3 = \{z, y\}$ *and* $S_4 = \{\top\}$.

S_1 *has one extension* λ_1^1 *with* $\lambda_1^1(x) = 1$. *This can be extended to* S_2 *in only one*

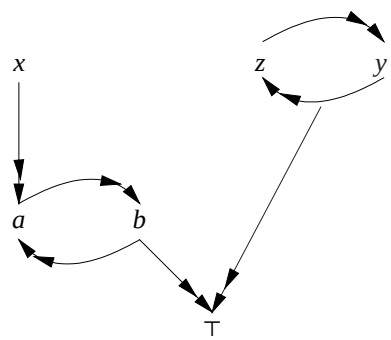


Figure 34:

way, namely λ_1^2 *with* $\lambda_1^2(a) = 0$ *and* $\lambda_1^2(b) = 1$.

However, because of $\top \in S_4$*, we cannot have* $\lambda_1^2(b) = 1$*, because it should be 0. Thus* $S_4 = \{\top\}$ *is toxic for* λ_1^1 *and* λ_1^2*. So we abandon* S_4 *for this sequence, (i.e.* $S_1 \twoheadrightarrow S_2 \twoheadrightarrow S_4$*) and get the extension* $\lambda^{1,2} = \lambda_1^1 \cup \lambda_1^2$*, with* $\lambda^{1,2}(x) = 1, \lambda^{1,2}(a) = 0$ *and* $\lambda^{2,1}(b) = 1$*. We ignore* $\top$.

Now let us look at S_3*. It has two extensions* λ_1^3, λ_2^3 *with* $\lambda_1^3(z) = 1$ *and* $\lambda_1^3(y) = 0$ *and* $\lambda_2^3(z) = 0$ *and* $\lambda_2^3(y) = 1$*. The extension* λ_1^3 *is not possible because* $z \twoheadrightarrow \top$*. So we have only the extension* λ_2^3.

The final extension for the $\top$*-net of Figure 34 is* $\lambda = \lambda_1^1 \cup \lambda_1^2 \cup \lambda_2^3$ *namely*

$$\lambda(x) = 1, \lambda(a) = 0, \lambda(b) = 1, \lambda(z) = 0 \text{ and } \lambda(y) = 1.$$

Note that we ignore $S_4 = \{\top\}$ *only for the evaluation of the path* $S_1 \twoheadrightarrow S_2 \twoheadrightarrow S_4$.

For the path $S_3 \twoheadrightarrow S_4$ *we do not ignore* $S_4 = \{\top\}$ *because there is an extension for* S_3 *which is OK.*

Thus the following λ' *is* not *an extension for* $S_1 \cup S_2 \cup S_3$*:*

$$\lambda'(x) = 1, \lambda'(a) = 0, \lambda'(b) = 1, \lambda'(z) = 1 \text{ and } \lambda'(y) = 0.$$

We can achieve the same result if we say that we abandon the attack $b \twoheadrightarrow \top$*, rather than the set* $\{\top\}$*. This view is better because we are not "touching" the toxic part in some cases, rather than deleting it.*

The next example B.3 shows how to do this.

Example B.3 *Let us do the option (iv) semantics for Figure 34 in a different way from the way done in Example B.2.*

We proceed as follows:

1. *We are given a* $\top$*-net* (S, R) *with* $\top \in S$*. In this case it is the net of Figure 34.*

2. *Replace every occurrence of* $\top \in S$ *by a new node letter* τ*, and get* (S_τ, R_τ)*, where*

$$\begin{aligned} S_\tau &= (S - \{\top\}) \cup \{\tau\} \\ R_\tau &= (R - \{(x, \top), (\top, y) | (x, \top), (\top, y) \in R\}) \cup \{(x, \tau), (\tau, y) | (x, \top), (\top, y) \in R\}. \end{aligned}$$

3. *(S_τ, R_τ) is a traditional network and so we seek and find all of its complete extensions (Caminada labelling) of the form λ_τ for which $\lambda_\tau(\tau) = 1$.*

   If such extensions exist then for each λ_τ let $\lambda_\top$ be the function on S obtained by

$$\begin{aligned} &\lambda_\top(x) = \lambda_\tau(x), \text{ for } x \neq \tau \\ &\lambda_\top(\top) = 1. \end{aligned}$$

   These will be the option (iv) extensions of Figure 34.

4. *If (S_τ, R_τ) has no exensions in λ in which $\lambda(\tau) = 1$, then let $\lambda_1, \ldots, \lambda_k$ enumerate all the extensions of (S_τ, R_τ) which give τ a value $\neq 1$.*

   Such extensions exist because we are dealing with a traditional network. Let us enumerate these extensions for our example (of Figure 34 with τ replacing $\top$).

   We have two such extesions, λ_1 and λ_2:

$$\begin{aligned} \lambda_1 : &\quad x = 1, a = 0, b = 1, \tau = 0, z = 1, y = 0 \\ \lambda_2 : &\quad x = 1, a = 0, b = 1, \tau = 0, z = 0, y = 1. \end{aligned}$$

5. *We know that the requirement that $\lambda(\tau) = 1$ is toxic, because of the attacks $b \twoheadrightarrow \tau$ and $z \twoheadrightarrow \tau$. Our remedy is to disconnect some of these attacks in order to get extensions. If we disconnect them all we certainly get such extensions, but we want to accommodate the attacks as much as we can.*

   Our possibilities are the following:

   (a) *Disconnect $z \twoheadrightarrow \tau$*
   (b) *Disconnect $b \twoheadrightarrow \tau$*
   (c) *Disconnect both $z \twoheadrightarrow \tau$ and $b \twoheadrightarrow \tau$*

   We give preference to disconnecting attacks emanating from points nearer the bottom (away from the top) of the figure. Thus (b) gets priority over (a).

   We also want to minimise the number of changes and so (c) has least priority.

   Disconnecting $b \twoheadrightarrow \tau$ gives us the extension

$$\lambda : \quad x = 1, a = 0, b = 1, \tau = 1, z = 0 \text{ and } y = 1.$$

   This yields a complete non-toxic extension for Figure 34.

6. *Note that we may wish to take into account the value of x under λ in our consideration of whether to disconnect an attack of $x \twoheadrightarrow \tau$. $\lambda(x)$ can be 1 or $\frac{1}{2}$ and we may decide not to fix λ (by disconnecting $x \twoheadrightarrow \tau$) if $\lambda(x) = 1$. If we do indeed make this decision then we would not disconnect $b \twoheadrightarrow \tau$ in our example and in such a case the network of Figure 34 will have no extensions!*

Remark B.4 *Example B.2 also shows that the result (semantics) we get for this option (iv): non-toxic truth intervention, is different from the result we get from option (ii), the counter attack view. According to option (ii), $\top$ counter attacks all its attackers and so we get the extension λ_c*

$$\lambda_c(x) = 1, \lambda_c(a) = 0, \lambda_c(b) = 0, \lambda_c(\top) = 1, \lambda_c(z) = 0 \text{ and } \lambda_c(y) = 1.$$

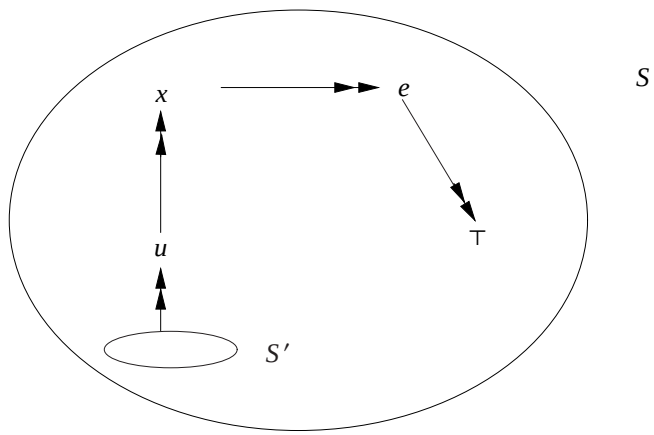


Figure 35:

Remark B.5 *The perceptive reader may ask why are we even considering the idea of using maximal non-toxic extensions? After all, did we not say that instantiating* (S, R) *for* $y \in S$ *with* $y = \top$ *amounts to looking for extensions in which* $y =$ *"in". So if there are no such extensions, then the straightforward answer is that there are no extensions! Why suddenly claim* $y = \top$ *is toxic and let us ignore it? On the one hand, we are interested in* y *and want it "in" and on the other hand when we cannot do that we throw out of the network with that very same* y*, calling it toxic!*

The answer is that there are cases of networks with y *where we are not looking at the instantiation* $y = \top$ *as a search for extensions in which* $y =$ *"in". We are looking in such a network at* $y = \top \in S$ *as an instrument of intervention of forcing an "in" value of some other node* z *related to* y *at the object level! Since we have a purpose in this case, then if our instrument does not work we need to find an alternative way.*[8]

Consider the network (S, R) *of Figure 35. Let* (S', R') *be the network obtained from* (S, R) *by deleting the nodes* $\{e, \top\}$*. Consider an intervention into* (S', R') *intending to enforce the node* x *to be "in". The nodes* $\{e, \top\}$ *added to* (S', R') *with* $e \twoheadrightarrow \top$ *are intended to achieve this purpose, in Figure 35.*

In this figure, since e *attacks* $\top$ *it must be "out", not because it attacks* $\top$ *but because some node with value "in" is attacking it. So* x *must be "in", because it is the only the attacker of* e*. We have used* $\top$ *and* e *in* (S, R) *to force* x *to be 'in" by the object level geometry of* (S, R)*. This means that only complete extensions of* (S', R') *with* $x =$ *"in" are acceptable. If, because of this intervention, we cannot have an extension, then*

[8]The Thesis [27] and the paper [26], deal with intervention. Suppose we have a network (S, R), and an element a in S. We want to intervene and force a to have value $e \in \{0, \frac{1}{2}, 1\}$. We can do that by adding to (S, R) some new point x with the following attack pattern:

1. For forcing a to be "out", let x attack a.
2. For forcing a to be "undecided" let x attack a and attack itself.
3. For forcing a to be "in" (if possible), let a attack x and let x attack $\top$ (we have to add $\top$ as well as x).

we can say that our attempt (intervention) fails, or in our terminology, is toxic, and consider giving it up, and we need to look for alternatives. Whatever reason we had for wanting x to be "in", must now be serviced by defining another complete extension in some other way. The way we define the extension depends on the purpose. Our purpose in this paper is connected with the soundness of the attack formations to be presented in Appendix C.2. So our definitions lead towards that purpose.

We are dealing here with a new type of instrument for defining extensions in the case where the present of $\top$ is toxic!

We are going to need to define priority on the set of elements attacking $\top$. This will be done in terms of their distance from the top of the network. To do this we need to use the notions of Strongly Connected Components. The next series of definitions (Definition B.6 to Definition B.8) deals with this.

Definition B.6 *Let (S, R) be an argumentation network. We define (S^*, R^*) the network of* strongly connected components *(SCC) derived from (S, R), following [20].*

1. *A subset $E \subseteq S$ is an SCC iff the following holds:*
   (a) $S \neq \emptyset$
   (b) *For any $x, y \in S$, there exists a sequence $z_1, \ldots, z_{k+1}$ such that $z_1 = x, z_{k+1} = y$ and for $1 \leq i \leq k$ we have $z_i R z_{i+1}$ holds.*
   (c) *E is maximal w.r.t. property (b).*
2. *Let S^* be the set of all SCC subsets of S. Define $E_1 R^* E_2$ on S^* iff for some $x_1 \in E_1$ and $x_2 \in E_2$ we have $x_1 R x_2$.*
3. *For $x \in S$, let x^* be the SCC to which it belongs.*

Lemma B.7 *Let (S, R) be a network and let (S^*, R^*) be its associated SCC network. Then*

1. *Any two distinct SCC sets are disjoint.*
2. *For any $x \in S$, there is a unique x^* for which it belongs.*
3. *R^* is well defined on S^* and is acyclic.*

Proof Easy. See [20]. ■

Definition B.8 *Let (S, R) be a network and let (S^*, R^*) be its derived SCC network. Let $E \in S^*$. We define the notion of "E is of level (k, n)" as follows:*

1. *E is of level (1,1) if there does not exist an $E' \in S^*$ such that $E' R^* E$. Think of the level index (k, n) as k is the minimal R^* distance from the top nodes in (S^*, R^*) and n is the maximal distance. The top nodes are distance 1.*
2. *E is of level $(k + 1, n + 1)$ if k is the minimal m of the level (m, n) of any E' such that $E' R^* E$, and n is the maximal such n.*

3. *Since each $x \in S$ is a member of a unique E, we can define a level (k, n) for each $x \in S$. It is the level of the E containing it.*

Example B.9 *Consider Figure 34. Then $\{x\}$ and $\{z, y\}$ are of level (1,1). $\{a, b\}$ is of level (2,2) and $\{\top\}$ is also of level (2,3).*

Definition B.10 *Let (S, R) be a $\top$-net, that is (S, R) is an argumentation network with a special $\top \in S$, with $\top$ not attacking itself. We want to define the non-toxic truth intervention semantics for it (option (iv) semantics).*

We are going to give the algorithm for finding all complete extensions for (S, R) in the form of Caminada labellings λ, $\lambda : S \mapsto \{0, \frac{1}{2}1\}$, with $\lambda(\top) = 1$.

Step 1: *Let τ be a letter disjoint from $\top$. Let (S_τ, R_τ) be $(S(\top/\tau), R(\top/\tau))$, where $A(x/y)$ is the result of substituting y in all occurrences of x in $A(x)$, where y is a completely new letter to A.*

Step 2: *(S_τ, R_τ) is a traditional argumentation network and has complete extensions. Let Λ be the set of all such extensions. This set is non-empty. Let Λ_τ be the subset of all extensions $\lambda \in \Lambda$ such that $\lambda(\tau) = 1$. Λ_τ may be empty.*

For each $\lambda \in \Lambda_\tau$, let $\lambda_\top$ be defined by $\lambda_\top(x) = \lambda(x)$, for $x \neq \tau$, $\lambda_\top(\top) = \lambda(\tau) = 1$.

Step 3: *If $\Lambda_\tau \neq \emptyset$ then let $\lambda_\top = \{\lambda_\top | \lambda \in \Lambda_\tau\}$ be declared as the set of all the extensions of (S, R). If $\Lambda_\tau = \emptyset$, then proceed to step 4.*

Step 4: *Let T_0 be the set of all $x \in S$ such that x attacks $\top$, i.e. $x \twoheadrightarrow \top$ is in R. Let T_1 be the set of all $y \in S$ such that $\top \twoheadrightarrow y \in R$.*

We want to assume that $T_1 = \emptyset$. This is possible to do because we can move to the network (S_∞, R_∞) where $S_\infty = S \cup \{\infty\}$, where ∞ is a new point such that $\infty \notin S$ and $R_\infty = (R - \{\top \twoheadrightarrow y | y \in T_1\}) \cup \{\infty \twoheadrightarrow y | y \in T_1\}$.

In other words, we ensure that $y \in T_1$ ends up "out" because it is attacked by ∞ which is "in" (not being attacked by anything) and we disconnect any attacks emanating from $\top$. Any extension λ found for (S_∞, R_∞) will yield an extension for (S, R), when restricted to S.

So we can assume now that $\top$ does not attack anything. We now proceed to find extensions for (S, R). Each such $x \in S$ has a unique level (k, n) associated with it as defined in Definition B.8.

Define a priority $\geq$ ordering on nodes $x, y \in S$ by

- *$x \geq y$ iff $n_x \geq n_y$ and if $n_x = n_y$ then $k_x \geqq k_y$*

where the level of x is (k_x, n_x) and the level of y is (n_y, k_y).

Step 5: *We assume that we have (S, R), where $\top \in S$, $\top$ does not attack anything, and when we look at (S_τ, R_τ) then all of its extensions $\lambda \in \Lambda$ satisfy $\lambda(\tau) \neq 1$, i.e. $\lambda(\tau) = \frac{1}{2}$ or =1.*

Rather than declare that (S, R) has no extnesions, we want to salvage some extensions out of Λ by declaring the role of $\top \in S$ in some cases to be toxic!

We can now consider T_0. T_0 can be divided for each $\lambda \in \Lambda$, into $T_0 = T^{\lambda}_{0,\frac{1}{2}} \cup T^{\lambda}_{0,1} \cup T^{\lambda}_{0,0}$ where

$$T^{\lambda}_{0,\frac{1}{2}} = \{x \in T_0 | \lambda(x) = \tfrac{1}{2}\}$$
$$T^{\lambda}_{0,1} = \{x \in T_0 | \lambda(x) = 1\}$$
$$T^{\lambda}_{0,0} = \{x \in T_0 | \lambda(x) = 0\}$$

We know for sure that either $T^{\lambda}_{0,1} \neq \emptyset$ or if $T^{\lambda}_{0,\frac{1}{2}} \neq \emptyset$.

Let us adopt the policy that if for some x attacking $\top$ we have $\lambda(x) = 1$ then we give up on λ.[9]

So we are considering the case where all attackers x of $\top$ have value $\lambda(x) = 0$ or $\lambda(x) = \frac{1}{2}$.

If we disconnect the attacks emanating from $T^{\lambda}_{0,\frac{1}{2}}$ onto $\top$ we get an extension λ for (S, R_λ) where

$$R_\lambda = R - \{x \twoheadrightarrow \top | x \in T^{\lambda}_{0,\frac{1}{2}}\}.$$

Now should we choose λ as a non-toxic extension for (S, R)? It is a question of priority. We have a priority relation $>$ on T_0. Let us extend it to priority on sets $T^{\lambda}_{0,\frac{1}{2}}$ by a lexicographic ordering first on the number of elements of the set $T^{\lambda}_{0,\frac{1}{2}}$ and second on the max on the value n_x of the index (k_x, n_x) of $x \in T^{\lambda}_{0,\frac{1}{2}}$. The winning $T^{\lambda}_{0,\frac{1}{2}}$ for this priority will yield the λs we call the extensions of (S, R).

C

The following sequence of Appendices C.1–C.4 contain technical results supporting the claims in Remark 2.10. The material was postponed to this Appendix because of its technical complexity.

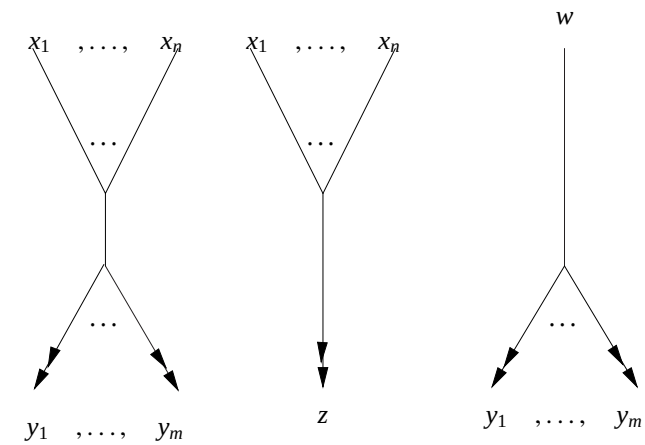
C.1 Conjunctive and disjunctive attacks

In Section 2.1 we introduced Boolean instantiation of argumentation networks. This allows for attacks of the form $(a \wedge b) \twoheadrightarrow (c \wedge d)$. The meaning of this is that if $(a \wedge b) = 1$ then $(c \wedge d) = 0$. We can write this as $\{a, b\} \twoheadrightarrow \{c, d\}$ and the meaning is that if both $a = b = 1$ then one of c or d equals 0.

This requires the study of conjunctive and disjunctive attacks. This is the task of Appendix C.1.

We recall concepts from [3].

[9] Note that in examples B.2 and B.3 we did fix the net of Figure 34. According to our present policy, we would not do that and declare the network of Figure 34 as having no extensions. On the other hand, if we wish to always have complete extensions to any $\top$-net (S, R) then we ignore $T_{\lambda,\frac{1}{2}}$ and proceed to fix λ.



(a) $X \twoheadrightarrow Y$ (b) $X \twoheadrightarrow z$ (c) $w \twoheadrightarrow Y$

Figure 36:

Definition C.1

1. *A conjunctive-disjunctive argumentation network, (CD-network) has the form* $(S, \mathbb{R})$ *where* S *is a finite set of arguments and* $\mathbb{R} \subseteq 2^S \times 2^S$ *is a relation between subsets. When* $X\mathbb{R}Y$ *holds between* $X, Y \subseteq S$*, we say that conjunctive* X *attacks* Y *disjunctively, or* X *CD-attacks* Y*. Figure 36 (a), (b), (c) shows our graphical notation for this notion.*

   We also write $X \twoheadrightarrow Y$ *for* $X\mathbb{R}Y$*,* $X \twoheadrightarrow z$ *for* $X\mathbb{R}\{z\}$ *and* $w \twoheadrightarrow Y$ *for* $\{w\}\mathbb{R}Y$*.*

2. *We require* $\mathbb{R}$ *to satisfy:* $X \twoheadrightarrow Y$ *and* $X' \supseteq X$ *and* $Y' \supseteq Y$ *imply* $X' \twoheadrightarrow Y'$*.*

3. *We say a node* $z \in S$ *is indirectly attacked by* $X \subseteq S$ *if for some* Y $X\mathbb{R}Y$ *and* $z \in Y$*.*

Definition C.2 (Kleene 3 valued logic) *Kleene 3-valued logic has 3 values* $\{0, 1, \frac{1}{2}\}$ *and has the truth table in Figure 37. The language has connectives* $\wedge, \vee, \neg, \rightarrow$ *and atomic propositions. Read the values as 1="in", 0 = "out", and* $\frac{1}{2}$ *= "undecided", see [19].*

Definition C.3

1. *Let* $(S, \mathbb{R})$ *be a CD network as in Definition C.1. Let* $\lambda : S \mapsto \{0, 1, \frac{1}{2}\}$ *be an assignment of values to the elements of* S*, pretending these are atomic logic propositions. Let* $X \subseteq S$*. Extend* λ *onto* X *by letting* $\lambda(X) = \lambda(\bigwedge_{x \in X} x)$*.*

   Thus

   $\lambda(X) = 1$ *if* $\lambda(x) = 1$ *for all* $x \in X$
   $\lambda(X) = 0$ *if for some* $x \in X, \lambda(x) = 0$
   $\lambda(X) = \frac{1}{2}$*, otherwise (i.e.* $\lambda(x) > 0$ *for all* $x \in X$ *and for some* $x \in X, \lambda(x) = \frac{1}{2}$*).*

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$
1	1	0	1	1	1
$\frac{1}{2}$	1	$\frac{1}{2}$	$\frac{1}{2}$	1	1
0	1	1	0	1	1
1	$\frac{1}{2}$	0	$\frac{1}{2}$	1	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	0	1	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
0	0	1	0	0	1

Figure 37:

2. *A CD-extension for* $(S, \mathbb{R})$ *is a function* $\lambda : S \mapsto \{0, 1\frac{1}{2}\}$ *satisfying the following conditions:*

   (a) *If* $\{z\}$ *is not indirectly attacked by any* X *then* $\lambda(z) = 1$.

   (b) *If* $\lambda(x) = 1$ *for all* $x \in X$ *and* $X \twoheadrightarrow Y$ *holds then for some* $y \in Y, \lambda(y) = 0$. *(I.e. if* $\lambda(X) = 1$ *and* $X \twoheadrightarrow Y$ *then* $\lambda(Y) = 0$.*)*

   (c) *Assume* Y *is such that for every* X *such that* $X \twoheadrightarrow Y$ *there exists an* $x \in X$ *such that* $\lambda(x) = 0$. *Then for all* $y \in Y$ *we have* $\lambda(y) = 1$. *(I.e. if* $\lambda(X) = 0$ *for all the attackers of* Y *then* $\lambda(Y) = 1$.*)*

   (d) *For any* $Y \subseteq S$, *(d1) and (d2) imply (d3).*

      (d1) *For every* X *such that* $X \twoheadrightarrow Y$ *we have that for some* $x \in X, \lambda(x) < 1$.

      (d2) *For some* X *such that* $X \twoheadrightarrow Y$ *we have that for all* $x \in X, \lambda(x) > 0$ *and for some* $x \in X, \lambda(x) = \frac{1}{2}$.

      (d3) *For some* $y \in Y, y = \frac{1}{2}$ *and for all* $y \in Y, \lambda(y) > 0$.

      (I.e. for all the attackers X *of* Y, $\lambda(X) \in \{0, \frac{1}{2}\}$ *and for at least one attacker* $X_0, \lambda(X_0) = \frac{1}{2}$ *then* $\lambda(Y) = \frac{1}{2}$.*)*

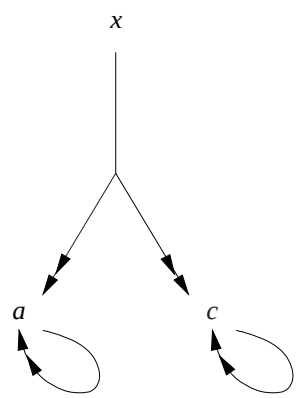
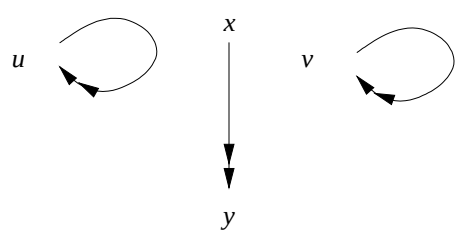


Figure 38:



Instantiate $u = a, v = c$ and $y = a \wedge c$

Figure 39:

Remark C.4

1. *Let $(S, \mathbb{R})$ be a CD-network as in Definition C.1. We ask whether there exist CD-extensions for it. To help us understand the situation, let us look at a figure which might make us think that the answer is negative. Consider an extension for the network of Figure 38.*

   Any such extension λ would require that $\lambda(x) = 1$ and either $\lambda(a) = 0$ or $\lambda(c) = 0$. But the fact that also $a \twoheadrightarrow a$ and $c \twoheadrightarrow c$ forces $\lambda(a) = \lambda(b) = \frac{1}{2}$. The values cannot be in $\{0, 1\}$!

   The perceptive reader might ask: we are interested in Boolean instantiations of traditional networks. Can Figure 38 be obtained as such an instantiation? If not then the use of CD-networks is an overkill. The answer is yes, see Figure 39

   If we write the equations for this figure we get

$$\begin{aligned} u &= 1 - u \\ v &= 1 - v \\ y &= 1 - x \\ x &= 1 \end{aligned}$$

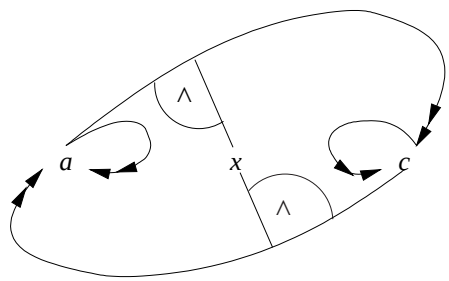


Figure 40:

Substituting the instantiation we get

$$a = 1 - a$$
$$c = 1 - c$$
$$a \wedge c = 0$$

There is no solution in $\{0, 1, \frac{1}{2}\}$.

This line of reasoning, however, contains a fallacy. Before the instantiation, u for example, was attacked only by itself, so the equation for it was $u = 1 - u$. *After the instantiation of* $u = a$ *and* $y = a \wedge c$, *a was attacked both by itslef and indirectly also by x. Thereofre a new equation should be written for the new situation of Figure 38. We cannot just substitute the instantiation in the old equations. We are not reading Figure 38 correctly.*

We need to modify our point of view. The following discussion is a conceptual analysis seeking a new point of view:

We begin with a slightly modified point of view which we call the RCD view (the restricted CD view). We note that $x \twoheadrightarrow \{a, c\}$ *actually implies* $\{x, a\} \twoheadrightarrow c$ *and* $\{x, c\} \twoheadrightarrow a$.

If we understand $x \twoheadrightarrow \{a, c\}$ *as meaning the conjunction of the above two attacks then Figure 38 becomes Figure 40.*

Since x is not indirectly attacked we have $\lambda(x) = 1$ *and since a and c are each self attacking we get* $\lambda(a) = \lambda(c) = \frac{1}{2}$.

We would also have according to Definition C.3 that $\lambda(\{x, a\}) = \lambda(\{x, c\}) = \frac{1}{2}$.

2. *It seems then, that if we adopt the suggestion in (1) above and understand* $x \twoheadrightarrow \{y_1, \ldots, y_k\}$ *from the RCD point of view, as the set of attacks* $\{x, y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k\} \twoheadrightarrow y_i$ *for* $i = 1, \ldots, k$, *then all we need to define is the concept of attacks of the form* $X \twoheadrightarrow z$. *Attacks of the form* $X \twoheadrightarrow Y$ *are reducible to the form* $X \twoheadrightarrow z$. *Such joint attacks can be simulated within traditional Dung networks using auxilliary points, as shown in [3].*

   The attack formation $\{x_1, \ldots, x_n\} \twoheadrightarrow z$ *of Figure 36(b) becomes the formation of Figure 41. The auxilliary points are* $y_1, \ldots, y_n, y$. *The reader should note*

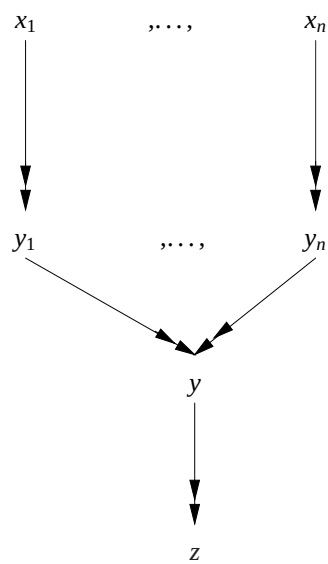


Figure 41:

that the auxiliary points used must be completely new to the rest of the network and be associated with $\{x_1, \ldots, x_n, z\}$ only. Any other attack formation, say $\{a_1, \ldots, a_m\} \twoheadrightarrow b$ will its own completely new and disjoint auxiliary points. If we do not observe this restriction and re-use existing points as auxiliary points we get wrong results, as Remark C.6 shows.

3. *The RCD point of view is not the best we can offer. There is a better one, already mentioned in the 2009 paper [3]. In Section 4.3 of that paper, we discussed what we call "flow argumentation networks". To explain it simply for our case, in Figure 36(c) the node w transmits attacks to all the nodes $y_1, \ldots, y_m$ (the attack "flow" emanating from w) but expects only at least one of them to succeed. Thus applying this "flow" view to Figure 38 we have that x attacks a and x attacks c but expects at least one to succeed. Since $x = 1$ and $a = \frac{1}{2}$, a will become 0. Similarly c will become 0. Thus we get two extensions*

$$x = 1, a = 0, c = \tfrac{1}{2}$$

and

$$x = 1, a = \tfrac{1}{2}, c = 0$$

We shall address this point of view, which we call the FCD-view in Appendix C2. More details in Section 4.3 of [3].

Remark C.5 *It would be instructive to compare our RCD disjunctive attack notion with the attacks notion of Nielsen and Parsons [18] notion of joint and disjunctive attacks. Call it the NP view. Let us look again at Figure 36(c).*

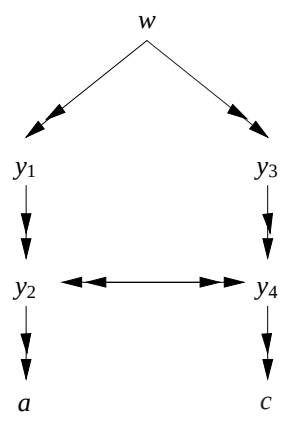


Figure 42:

We have

$$w \twoheadrightarrow Y = \{y_1, \ldots, y_m\}.$$

Let us simplify and consider $w \twoheadrightarrow \{a, c\}$. *We do our comparison for this case but before that let us summarise our options:*

We have the following four notions for $X \twoheadrightarrow Y$, *where*

$$X = \{x_1, \ldots, x_n\} \text{ and } Y = \{y_1, \ldots, y_m\}.$$

1. *Gabbay 2009 [3], CD-view:*

   If $\bigwedge_i \lambda(x_i) = 1$ *then* $\bigvee_i \lambda(y)i) = 0$.

   This view allows us to generate networks like in Figure 38.

2. *The equational view, which is not the same as the CD-view. This view is connected with instantiation networks and may not be able to generate networks like Figure 38. This view will require us to solve the equation* $(\bigwedge_j)y_j) \leftrightarrow \neg \bigwedge_i x_i$.

3. *Gabbay alternative as in (1) above RCD-view:*
   $w \twoheadrightarrow \{a, c\}$ *means* $\{w, a\} \twoheadrightarrow c$ and $\{w, c\} \twoheadrightarrow a$.

4. *Nielsen and Parsons' 2007 [18], the NP-view:*
   $w \twoheadrightarrow \{a, c\}$ *means* $w \twoheadrightarrow a$ or $w \twoheadrightarrow c$.

   (The Nielsen and Parsons' definition of $X \twoheadrightarrow Y$ *is that for some* $y \in Y, X \twoheadrightarrow y$*).*

The implementation of the NP-view for $w \twoheadrightarrow \{a, c\}$ *is in Figure 42.*

We need a semantics which will not allow for $\{y_2, y_4\}$ *to be both undecided.*

To complete our full comparison of the above four approaches, consider Figure 43.

This figure says that $\{a, b\} \twoheadrightarrow \{a, b\}$.

The CD approach and the equational approach offer no stable $\{0, 1\}$ *extensions. The RCD approach reduces* $\{a, b\} \twoheadrightarrow \{a, b\}$ *into*

$$a \wedge b \twoheadrightarrow a$$

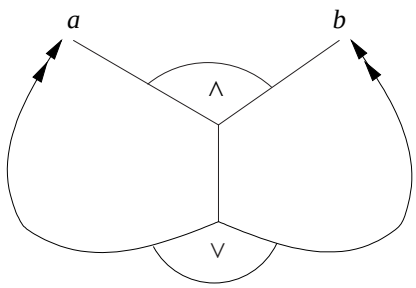


Figure 43:

and

$$a \wedge b \twoheadrightarrow b$$

and has no stable $\{0, 1\}$ *extensions either.*

However, the NP approach has two stable $\{0, 1\}$ *extensions*

$$a = 1, b = 0$$

and

$$a = 0, b = 1$$

This is because if $b \twoheadrightarrow a$ *or* $a \twoheadrightarrow b$ *then* $\{a, b\} \twoheadrightarrow \{a, b\}$ *holds. There is more discussion of the NP-approach in [3].*

Remark C.6 *If the joint attacks are done in two state networks, as defined and discussed in Definition 1.2 and Lemma 1.3, then we might think that it is much simpler to reduce joint attacks to single attacks by using existing points as auxiliary points. The attack*

$$a \wedge b \twoheadrightarrow c$$

can be reduced to the two attacks

$$\neg a \twoheadrightarrow \neg c$$
$$\neg b \twoheadrightarrow \neg c.$$

The reason for that equivalence can be seen by looking at the attack equationally. $a \wedge b \twoheadrightarrow c$ *means that* $c \equiv \neg(a \wedge b)$*. Therefore* $\neg c \equiv a \wedge b$ *or equivalently* $\neg c \equiv \neg(\neg a) \wedge \neg(\neg b)$ *which is the same equation for* $\neg a \twoheadrightarrow \neg c$ *and* $\neg b \twoheadrightarrow \neg c$*.*

In a network where for each x*,* $\neg x$ *is also present with* $x \twoheadleftarrow\!\!\twoheadrightarrow \neg x$ *(as we have in a 2-state network) then Figures 44 and 45 are equivalent.*

Notice the similarity between Figure 45 and Figure 41. If we let $y_1 = \neg x_1, \ldots,$ $y_n = \neg x_n$ *and* $y = \neg z$ *and change the attacks* $x_i \twoheadrightarrow y_i$ *into* $y \twoheadleftarrow\!\!\twoheadrightarrow z$*, the two figures become the same. Note that change the atatcks "*$\twoheadrightarrow$*" in Figure 41 into bidirectional attacks "*$\twoheadleftarrow\!\!\twoheadrightarrow$*' does not affect the job that Figure 41 does. We still have the same input/output relation between* $\bigwedge_i x_i$ *and* z*, namely* $\bigwedge x_i \twoheadrightarrow z$*.*

So we reduce the above attack $\wedge x_i \twoheadrightarrow z$ *to the single attacks* $\neg x_i \twoheadrightarrow \neg z, i = 1, \ldots, n$*.*

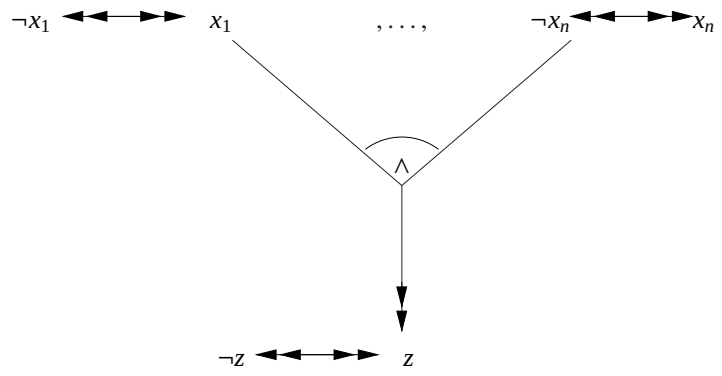


Figure 44:

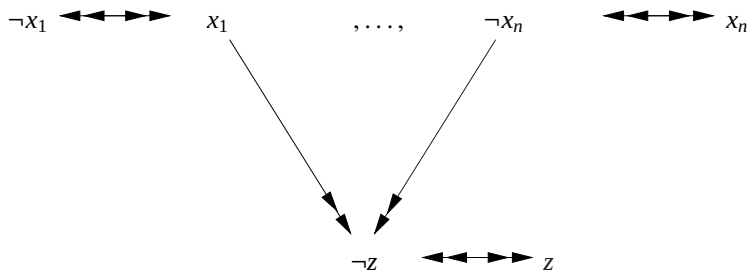


Figure 45:

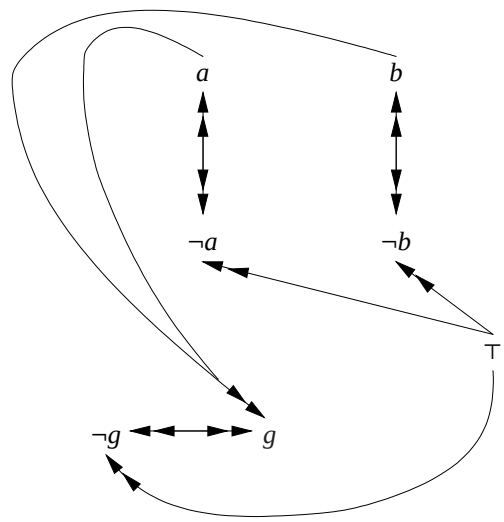


Figure 46:

To do this, however, by utilising the existing points $\{\neg x_i, z\}$ *as auxiliary points is a mistake. We must use only new, disjoint set of auxiliary points. Otherwise we get the wrong results.*

Consider Figure 46. In this figure $\top$ *is "in",* $\neg a = \neg b =$ *"out",* $\neg g =$ *"out" and since* $\{a, b\}$ *jointly attack g, g must also be out. This contradicts the notation implying that at least one of* $\{x, \neg x\}$ *must be in. If we ignore this restriction, we get that both g and* $\neg g$ *are "out". In this case think of* $\neg x$ *as just another* x'.

Figure 47 eliminates the joint attack $\{a, b\} \twoheadrightarrow g$ *by using* $\neg a, \neg b, \neg g$ *as auxiliary points, instead of using copletely new points. What we get is the wrong result.*

What we get is $\top =$ *"in",* $\neg a =$ *"out"* $= \neg b = \neg g$. $a = b = g =$ *in.*

Figure 47 eliminates the joint attacks using auxiliary points which are completely new, as shown in Figure 41.

We get the correct result, same as in Figure 46. We hae $\top = a = b =$ *"in". so* $x = y =$ *"out". So* $z =$ *"in" and so* $g =$ *"out".*

We note that it is for this reason that we introduce in Apendix C for the Boolean attack formations. These encapsulate the new auxiliary points inside the formation.

Note also that the presence of the $\{w, \neg w\}$ *pairs can allow us to possibly economise and use less auxiliary points, as Figure 49 shows. We economise by letting* $\neg x = a$ *and* $\neg y = b$.

There is no advantage, however, in economising. what is important is that joint attacks can be eliminated in a systematic way.

C.2 Boolean attack formations (BAF)

In this appendix, we generalise the instantiation sequence of the form of Figure 50 where x_i, y_j are atomic arguments and $\Psi(a_1, \ldots, a_n)$ is a Boolean formula in the ar-

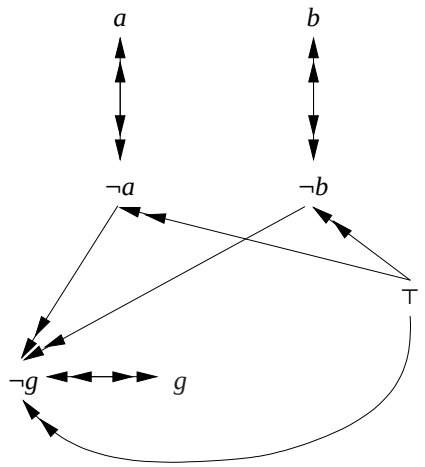


Figure 47:

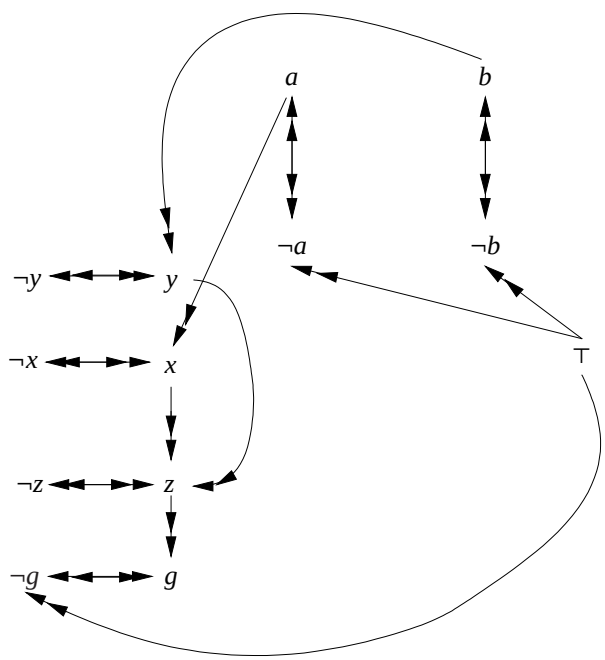


Figure 48:

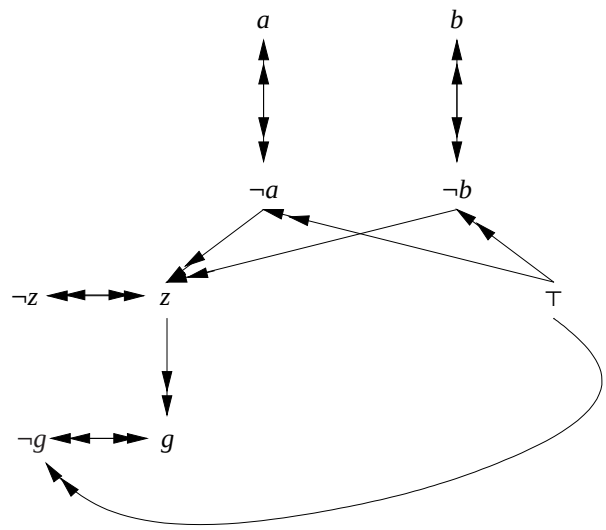


Figure 49:

guments $\{a_1, \ldots, a_n\}$. What we see in this figure is a substitution of some complex argumentation entity for the node z. Traditional abstract argumentation networks know how to handle attacks on atomic nodes z, they do not know how to deal with attacks on Boolean formulas. This Appendix C.2, replaces the formulas by attack formations and defines how to handle them. We thus define the notion of a Boolean Attack Formation with input and output nodes which can be substituted for nodes z. All attacks on z go into the input point of the formation and all attacks from z emanate from the output point of the formation replacing z.

We begin with the special case of Boolean attack formations designed to represent conjunction of formulas. Then we define formations to represent negation of formulas. Since negations and conjunctions can generate any formula of propositional classical logic, we will have attack formation representation for any classical propositional logic formula. The general definition, therefore, allows for general input/output formations.

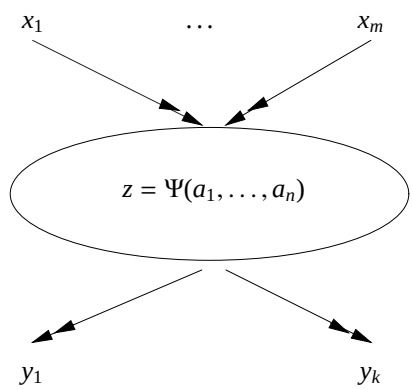
Definition C.7 *A Boolean attack formation $\mathbb{BF}$ has the form*

$$\mathbb{BF} = (S_1 \cup S_2, R, \Psi(S_2))$$

where $S_2 = \{a_1, \ldots, a_n\}$ and $\Psi(S_2)$ is a Boolean formula of Kleene 3 valued logic in the variables $\{a_1, \ldots, a_n\}$. We also write $\mathbb{BF} = \mathbb{BF}(a_1, \ldots, a_n)$.

The following holds:

1. *S_1 is a set disjoint from S_2 containing two special nodes among other additional nodes. There are* **in** *(an input node) and* **out** *(output node). S_1 is referred to as the set of auxiliary nodes. When several attack formation are involved we always assume that their sets of auxiliary node, including the in and out nodes are pairwise disjoint.*



z is instantiated as $I(z) = \Psi$.

Figure 50:

2. *We describe* $(S_1 \cup S_2, R, \Psi)$ *schematically in Figure 51.*

3. *Let* (S, R) *be an argumentation network and assume that* $\mathbb{BF}(a_1, \ldots, a_n)$ *is a subnetwork of* (S, R)*. We say that* $\mathbb{BF}$ *is legitimately embedded in* (S, R) *if the following holds:*

   (a) *The only elements of* $\mathbb{BF}$ *attacked from outside* $\mathbb{BF}$ *(by elements of* S*, which may include the elements* a_i *themeselves as attackers from* S*) are* $a_1, \ldots, a_n$ *and* **in** *(of* $\mathbb{BF}$*).*

   (b) **out** *of* $\mathbb{BF}$ *attacks only elements outside* $\mathbb{BF}$*.*

   The elements of $\mathbb{BF}$ *which are not in* $\{a_1, \ldots, a_n\}$ *appear only in* $\mathbb{BF}$ *(so* **in** *and* **out** *are labelled* **in**$(\mathbb{BF})$ *and* **out**$(\mathbb{BF})$*).*

4. *The following must hold for* $\mathbb{BF}(a_1, \ldots, a_n)$ *when embedded legitimately in any* (S, R)*, see Figure 52.*

   Let $\lambda : S \mapsto \{0, 1, \frac{1}{2}\}$ *be any extension of* (S, R)*. Then*

   (a) $\lambda(\mathbf{out}) = \Psi(\lambda(a_1), \ldots, \lambda(a_n))$.

   (b) *If none of* $\{a_1, \ldots, a_n\}$ *are attacked from outside* $\mathbb{BF}$ *or if all outside attackers* z *of* $a_1, \ldots, a_n$ *are out, i.e. have* $\lambda(z) = 0$*, then* $\lambda(\mathbf{in}) = \Psi(\lambda(a_1), \ldots, \lambda(a_n))$.

5. *If* $\Psi(\lambda(a_i)) = 1$ *then* $\lambda(\mathbf{in}) = 1$*, even if some* a_i *are attacked from outside* $\mathbb{BF}$*. It could be the case, however, that even though* $\lambda(\mathbf{in}) = 1$*, we have* $\Psi(\lambda(a_i)) \neq 1$ *because of outside attacks on* $\{a_i\}$*.*

Example C.8 *We show that we can find a* $\mathbb{BF}$ *for every formula of classical propositional logic. We show this by finding a* $\mathbb{BF}$ *for atomic* d*, for negation* $\neg d$ *and for*

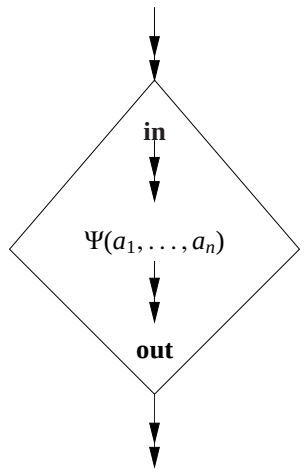


Figure 51:

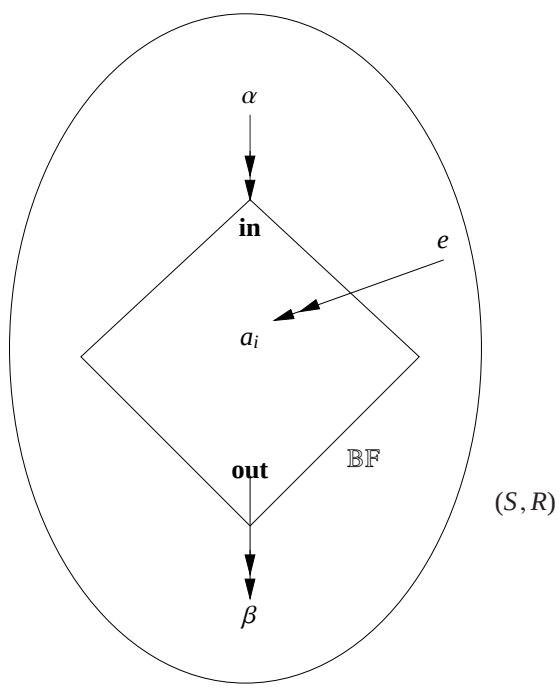


Figure 52:

$$\alpha \twoheadrightarrow d \twoheadrightarrow \beta$$

Figure 53:

$$\alpha \twoheadrightarrow \textbf{in} \twoheadrightarrow x \twoheadrightarrow d \twoheadrightarrow y \twoheadrightarrow \textbf{out} \twoheadrightarrow \beta$$

Figure 54:

conjunctions $a \wedge c$. The $\mathbb{BF}$ for arbitrary formulas can be done by legitimate substitutions of $\mathbb{BF}$s.

Consider the network of Figure 53.

This network can be replaced by Figure 54.

The attack formation of Figure 55 is involved where x, y are auxiliary points and $\Psi(d) = d$.

The $\mathbb{BF}$ for $\neg d$ can be obtained from the $\mathbb{BF}$ of d, (i.e. from Figure 54) by deleting the node x, i.e. allowing **in** *to attack d directly.*

The above two cases are simple but consider the more complex case of conjunction of Figure 56.

In this figure, α disjunctively attacks $\{a, c\}$ and $\{a, c\}$ conjunctively attacks β. Figure 56 is the same in meaning as Figure 53 instantiated by $d = a \wedge c$, namely Figure 57.

In this case it is not so simple to find a Boolean attack formation to do the same job. Namely we want Figure 58 to be realised as a traditional network.

The network of Figure 59 can do the job. It is a $\top$-net. We assume we are dealing with option (iv) of Appendix B, adopting the non-toxic approach.

Lemma C.9 *Figure 59 implements in $\top$-net non-toxic approach with auxiliary points the concept of disjunctive attacks.*

Proof We need to show the following for any λ.

1. If $\lambda(\alpha) = 1$, then for some i, $\lambda(a_i) = 0$ and $\lambda(w) = 0$.
2. If $\lambda(\alpha) = 0$, then unless attacked from outside the diamond, all $\lambda(a_i) = 1$ and $\lambda(w) = 1$.
3. If $\lambda(w) = \frac{1}{2}$, then $\lambda(a_i) < 1$. They are all 0 or $\frac{1}{2}$ and if some a_i is not successfully attacked from the outside then $\lambda(a_i) = \frac{1}{2}$. Thus $\lambda(w) = \frac{1}{2}$, unless some $\lambda(a_i) = 0$ in which case $\lambda(w) = 0$.

   Let us examine each case.

   Case 1: Assume $\lambda(\alpha) = 1$. Hence $\lambda(\overline{\alpha}) = 0$ and $\lambda(\overline{\overline{\alpha}}) = 1$. Hence $\lambda(v) = 0$ and $\lambda(x) = 0$. We know that $\lambda(e)$ must be 0, so one of its attackers must be 1.[10] It

[10] Note that in option (i) for the semantics, an attacker of $\top$ must be "out" not by virtue of it attacking $\top$ but because it must have an attacker which is "in".

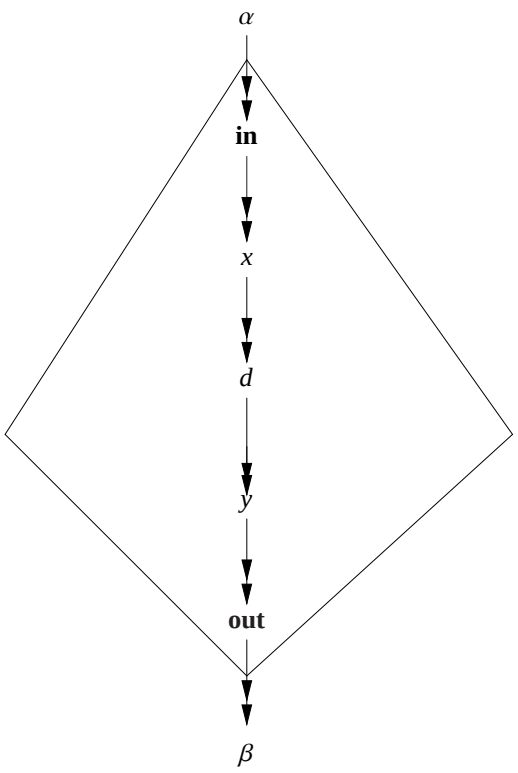


Figure 55:

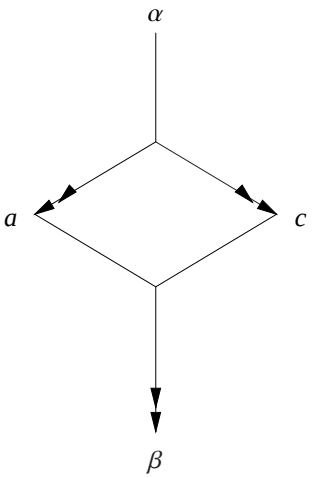


Figure 56:

$$\alpha \twoheadrightarrow (a \wedge c) \twoheadrightarrow \beta$$

Figure 57:

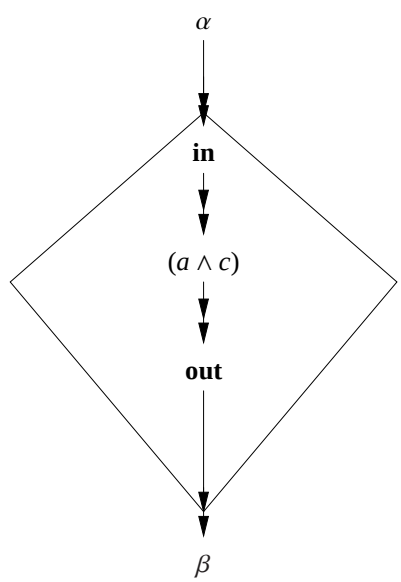


Figure 58:

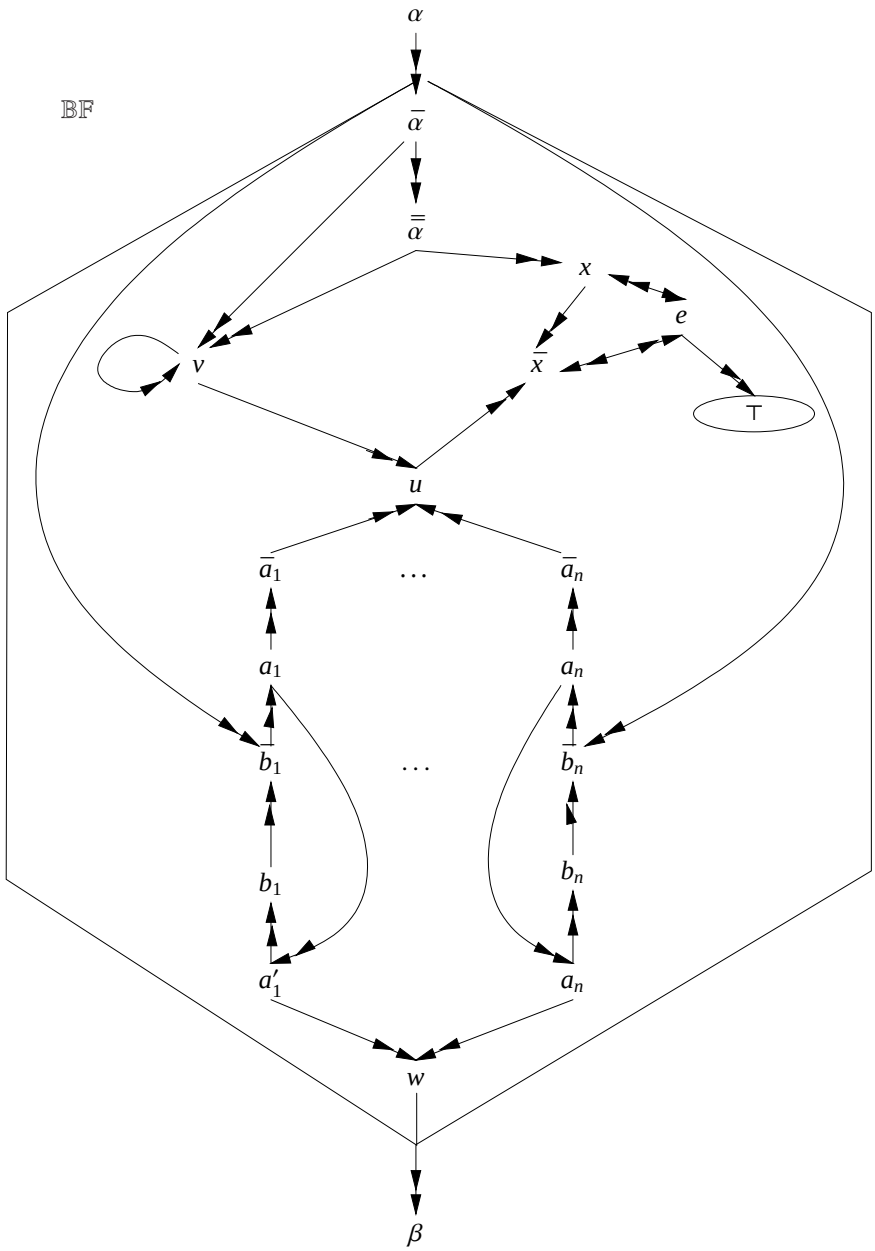

$\mathbb{BF}$
α
$\overline{\alpha}$
$\overline{\overline{\alpha}}$
x
e
v
$\overline{x}$
$\top$
u
$\overline{a}_1$
$\dots$
$\overline{a}_n$
a_1
a_n
$\overline{b}_1$
$\dots$
$\overline{b}_n$
b_1
b_n
a'_1
a_n
w
β

Figure 59:

cannot be x so it must be $\overline{x}$. If this is the case then $\lambda(u) = 0$ and then for some $i, \lambda(\overline{a}_i) = 1$.

Note that since $\lambda(\overline{\alpha}) = 0$, the loop $a_i \twoheadrightarrow a'_i \twoheadrightarrow b_i \twoheadrightarrow \overline{b}_i \twoheadrightarrow a_i$ is not broken. So a_i is attacked and so either the loop is resolved by $\lambda(a_i) = \lambda(b_i) = 1$ or by $\lambda(a_i) = \lambda(b_i) = 0$.

However, since $\lambda(\overline{a}_i) = 1$ we must have $\lambda(a_i) = 0$. In which case $\lambda(a'_i) = 1$ and $\lambda(w) = 0$.

Case 2: Assume $\lambda(\alpha) = 0$. Then $\lambda(\overline{\alpha}) = 1, \lambda(\overline{\overline{\alpha}}) = 0$. Hence $\lambda(v) = 0$ and also $\lambda(x) = 1$ because $\lambda(e)$ has to be 0. Therefore $\lambda(\overline{x}) = 0$. $\lambda(u)$ can be anything. It needs not be 1 or $\frac{1}{2}$ because $\lambda(x) = 1$ forces $\lambda(\overline{x}) = 0$.

Now since $\lambda(\overline{\alpha}) = 1$, we get $\lambda(\overline{b}_i) = 0$. Thus $a_1, \ldots, a_n$ are not attacked from any point in the inside. So unless a_i is attacked from the outside, we have $\lambda(a_i) = 1$. So if all $\lambda(a_i) = 1$ for all i, we get all $\lambda(a'_i) = 0$ for all i and so $\lambda(w) = 1$.

Case 3: Assume $\lambda(\alpha) = \frac{1}{2}$, then $\lambda(\overline{\alpha}) = \lambda(\overline{\overline{\alpha}}) = \lambda(v) = \lambda(x) = \frac{1}{2}$. Since $\lambda(e)$ must be 0, we look for an attacker of e whose λ value is 1. The two candidates are x and $\overline{x}$. $\lambda(x) = \frac{1}{2}$ implies $\lambda(\overline{x}) \neq 1$. Thus there is no way forward and it is obvious that the set $\{\top\}$ is toxic for the case of the input $\lambda(\alpha) = \frac{1}{2}$. We need to ignore it. See Appendix B, especially Definition B.10.

In this case we get that $\lambda(\alpha) = \lambda(\overline{\alpha}) = \lambda(\overline{\overline{\alpha}}) = \lambda(v) = \frac{1}{2}$.

We do not care what values the set $\{x, \overline{x}, e\}$ gets, for example it can get the values $\lambda(e) = 1, \lambda(x) = \lambda(\overline{x}) = 0$. The question is what is $\lambda(u)$? It cannot be 1 because $\lambda(v) = \frac{1}{2}$. Can it be 0? If $\lambda(u) = 0$, then for some $i, \lambda(\overline{a}_i) = 1$. Then in this case $\lambda(a_i) = 0$. Then $\lambda(a'_i) = 1, \lambda(b_i) = 0$, but then $\lambda(\overline{b}_i) = \frac{1}{2}$ because it is attacked by $\overline{\alpha}$ and $\lambda(\overline{\alpha}) = \frac{1}{2}$. So how can $\lambda(a_i) = 1$?

So the last possibility is $\lambda(u) = \frac{1}{2}$. Then $\lambda(\overline{a}_i) = \frac{1}{2}$ or 0 and for at least one i $\lambda(\overline{a}_i) = \frac{1}{2}$.

If $\lambda(\overline{a}_i) = 0$, then $\lambda(a_i) = 1$ so $\lambda(a'_i) = 0$, $\lambda(b_i) = 1$ and $\lambda(\overline{b}_i) = 0$. We know that for at least one i, we have $\lambda(\overline{a}_i) = \frac{1}{2}$. So $\lambda(a'_i) = \frac{1}{2}$.

Thus from the attackers of $w, \{a'_i\}$, at least one has value $\frac{1}{2}$ and the rest of the values are $\frac{1}{2}$ or 0. So $\lambda(w) = \frac{1}{2}$.

■

C.3 Instantiating with Boolean attack formations

This appendix prepares the technical ground for intantiating with Boolean formulas. We first turn any such formula into a Boolean attack formation (see Example C.8) and then instantiate with the resulting formations. To achieve that we need to define the notion of instantiation with $\mathbb{BF}$s.

We need to agree on some diagrammatic conventions. Consider Figure 60:

$$a \twoheadrightarrow b \twoheadrightarrow a \twoheadrightarrow b'$$

Figure 60:

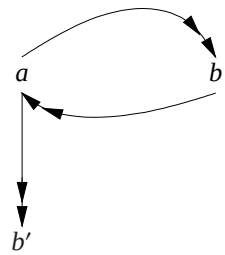


Figure 61:

In this figure the letter a appears twice. Is this a misprint and the second a should be a', or is the intention actually as depicted in Figure 61?

The answer is that sometimes figures can be very complex and for the purpose of simplification, a letter can be repeated. To be on the safe side a repeated letter can be encircled like in Figure 62.

The need for repetition comes from the notion of $\mathbb{BF}$. In Definition C.7, we see that any $\mathbb{BF}$ has two types of variables, the auxiliary variblaes $S_1^{\mathbb{BF}}$, unique to the $\mathbb{BF}$ containing it and disjoint from any other $\mathbb{BF}'$, and the variables $S_2^{\mathbb{BF}}$, which may be attacked by other $\mathbb{BF}$s which may share some of these ariables in $S_2^{\mathbb{BF}'}$. Consider, for example, Figure 63.

In this figure, $\{a, c\}$ mount jointly a disjunctive attack on $\{a, b, c\}$. See Appendix C1. If we do not want repetition of nodes, we write Figure 64.

However, if we want to implement Figure 64 using Boolean attack formations as discussed in Appendix C2, we get Figure 65, with auxiliary nodes $a_1^*, c_1^*, z_1^*, w_1^*$ and the auxiliary nodes of Figure 59 for $\{a, b, c\}$.

So be mindful that the following definitions allow for repetions in the figures.

Definition C.10

1. *Let $\mathbb{BF}_1$ and $\mathbb{BF}_2$ be two Boolean attack formations with* $\mathbf{in}_1$ *and* $\mathbf{out}_1$ *and* $\mathbf{in}_2$ *and* $\mathbf{out}_2$. *We assume that the auxiliary nodes of these formations are disjoint. Then Figure 66 describes the resulting formation for the attack of $\mathbb{BF}_1$ on $\mathbb{BF}_2$.*

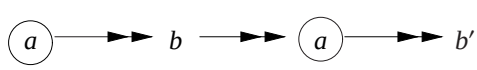


Figure 62:

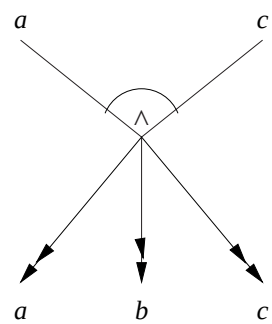


Figure 63:

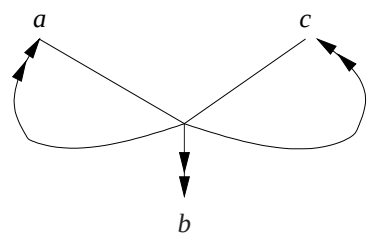


Figure 64:

2. *Let (S, R) be an arumentation network and let I be a function associating with each $x \in S$, a Boolean attack formation $I(x) = \mathbb{BF}_x$, with $\mathbf{in}_x, \mathbf{out}_x, S_x, R_x$. Assume that all the auxiliary nodes sets of $\mathbb{BF}_x$ for $x \in S$ are pairwise disjoint and disjoint from S. Then the result of the instantiation (S_I, R_I) is the following:*

$$\begin{array}{lcl} S_I & = & \bigcup_{x \in S} S_x \\ R_I & = & \bigcup_{x \in S} R_x \cup \{\mathbf{out}_x \twoheadrightarrow \mathbf{in}_y | x \twoheadrightarrow y \in R\}. \end{array}$$

3. *The semantics for such instantiation is taken to be option (iv) (the non-toxic truth intervention) semantics of Appendix B.*

C.4 Instantiating with classical propositional wffs using Boolean attack formations

In this Appendix C.4, we backup the comments made in Remark 2.10. We are given a finite instantiated network (S, R, I). I is an instantiation into classical propositional logic and we can assume that all wffs involved are built up from the set of atoms $Q = \{q_1, \ldots, q_n\}$ using $\wedge$ and $\neg$. We want to identify the Caminada extensions for (S, R), arising from the instantiation I. We know that for each $x \in S$, the wff $I(x) = \Phi_x$ is a formula of a logic which has models $\mathbf{m}$. We know that if we go through all the models $\mathbf{m}$ and define

$$\lambda_{\mathbf{m}}(x) = \text{def. } \mathbf{m}(\Phi_x)$$

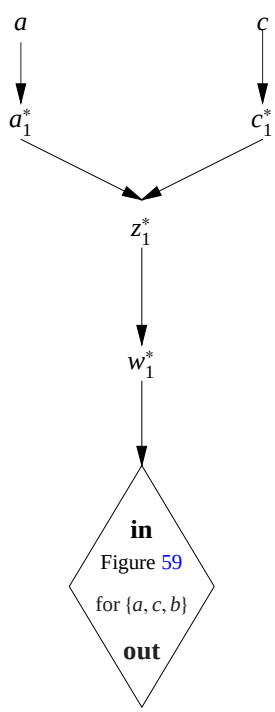


Figure 65:

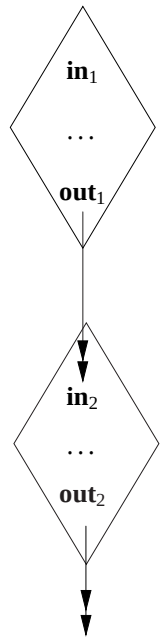


Figure 66:

then the legitimate Caminada extension of (S, R) among the set $\{\lambda_{\mathbf{m}}\}$ will be the set of all extensions of (S, R, I). We can do that, however, we do not want to use the logic, we want to use purely syntactical means. By the results in Appendix C.2, every formula Φ of classical propositional logic can be represented by a Boolean attack formation $\mathbb{BF}(\Phi)$. We can look at the instantiation I^* where $I^*(x) = \mathbb{BF}(\Phi_x)$ for $x \in S$ and consider the network $(S^*, R^*) = (S, R, I^*)$ as defined according to Definition C.10 of Appendix C.3.

The network (S^*, R^*) has extensions μ (the non-toxic extensions).

These extensions give values in $\{0, \frac{1}{2}, 1\}$ to the atoms of Q. These define a model $\mathbf{m}(\mu)$ to the formulas $\Phi_x, x \in S$. We look at $\{\lambda_{\mathbf{m}(\mu)}\}$ and the legitimate Caminada extensions from among all the extensions of (S, R, I).

Remark C.11 *Note that as a byproduct of our process of Remark 2.10 and Appendix C.4, we get argumentation semantics for classical propositioal logic. Any Φ is represented by $\mathbb{BF}(\Phi)$, whose extensions give the models of Φ.*

D Comparing Boolean instantiation with abstract dialectical framework (ADF).

We first introduce ADF from the original brilliant paper of Brewka and Woltran [2, 35], and then analyse ADF and explain why it is not suitable for us. See also [36].

Brewka and Woltran read condition ($\sharp$1 atomic) above (i.e the view: "$y_1, \ldots, y_n$ attack x", see Figure 19) not as an attack of y_i on x but as an acceptance condition, relating the $\{0, 1\}$ values of the y_i to the $\{0, 1\}$ value of x. The condition, according to the ADF view, is

$$x = 1 \text{ iff all } y_i = 0, \text{ (namely we take } I_{(\sharp 1 \text{ atomic})}(x) \text{ as } \bigwedge \neg y_i).$$

This is a brilliant shift in point of view and we now can generalise and put forward different conditions, say $I(x) = C_x(y_1, \ldots, y_n)$. We can now interpret $I(x)$ as the Brewka–Woltran condition for the acceptance of x. To achieve that we must restrict the variables in $I(x)$ in our network (S, R, I) to be the set Y_x of all the attackers in (S, R) of the node x.

So our networks will look like (S, R, I), where $I(x) = \Phi_x\{Y_x\}$.

Now we can regard our instantiation networks with this additional restriction possibly as an ADF and we might think that our problems are solved. We have given a meaning to (S, R, I), at least for the case of this additional restriction.

The answer is that this is not the meaning we want and need. We cannot get the general case of instantiations which we are studying, even with this additional restriction. Consider the simple two node network with nodes x and y and with x attacking y. Since x is not attacked, ADF can give it a truth value only, say value $\mathbf{t}$, and since x is the only attacker of y, ADF can give it a wff $\Phi(x)$ with x as the only variable of Φ. ADF will write the equalities

$$\begin{aligned} x &= \mathbf{t} \\ y &= \Phi(x). \end{aligned}$$

While in comparison, we would write the equalities

$$\begin{aligned} x &= \mathbf{t} \\ \Phi(x) &= \neg\mathbf{t} \end{aligned}$$

Note however, that we can get ADF as a special case of our instatiation if we take an arbitrary network with nodes S and no attacks whatsoever, i.e. $R = \emptyset$.

If we substitute now for each x in S a wff Φ_x, we get an ADF.

The fact is that ADF is not doing argumentation but is really doing logic programming. We are departing from the principle ($\sharp$1) and we are not regarding $I(y_1), \ldots, I(y_m)$ as attacking $I(x)$.

If we take the ADF view then we are not doing argumentation any more. We are not really instantiating x to be $I(x)$. We can no longer connect with, e.g., ASPIC+ or any other instantiation papers. We are playing a different game!

This is not a criticism of ADF; all we are saying is that the ADF approach is not compatible with our approach. We think highly enough of ADF to mention it here and say that we cannot use it. Note for example that with the ADF restriction, R is redundant. This is a warning sign for us. All we really have in ADF are logic programming clauses of the form

$$x \text{ if } I(x)$$

and we have

$$xRy \text{ iff } y \text{ appears in } I(x).$$

Thus in ADF R is not an attack relation but an “occur" relation. Brewka and Woltran are careful; they call it “link" relation.

Our view about ADF becomes clearer when we consider it in the context of the equational approach [1, 4, 6, 7, 36].

Given (S, R), we consider the network as generating equations of the form $Eq_{\max}$:

$$x = 1 - \max(\mathrm{Y_x})$$

where $x \in S$ and $Y_x = \{y | yRx\}$.

The solution of these equations in $[0, 1]$ yield all the Dung extensions, where with each $\mathbf{f}$ we associate a Caminada labelling $\lambda(\mathbf{f})$ as follows:

$$\begin{aligned} \lambda(\mathbf{f}) &= \text{ in, if } \mathbf{f}(x) = 1 \\ \lambda(\mathbf{f}) &= \text{ out, if } \mathbf{f}(x) = 0 \\ \lambda(\mathbf{f}) &= \text{undecided, if } 0 < \mathbf{f}(x) < 1. \end{aligned}$$

Let us now depart from the above equations in two ways.

1. Let us have possibly different continuous functions $h_x(Y_x)$ associated with each $x \in S$.

2. Let $h_x(Y_x)$ be a Boolean function in Y_x

The solutions $\mathbf{f}$ of the above system of equation (call it Eq_B) would yield us a Boolean ADF. See [35, p. 804] and [36].

The reader should note the generality of the equational approach. We can choose any family of continuous functions. We can choose T-norms which generalise the classical connectives and using T-norms study numerical ADF (T-norm). However, this is not the place to elaborate, but we leave this to a future paper.

Let us at this point quote from paper [35]:

> Begin quote from [35].
>
> **Definition 1.** An *abstract dialectical framework* is a tuple $D = (S, L, C)$ where
>
> - S is a set of statements (positions, nodes),
> - $L \subseteq S \times S$ is a set of links,
> - $C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{par(s)} \to \{\mathbf{t}, \mathbf{f}\}$, one for each statement s. C_s is called acceptance condition of S
>
> In many cases it is convenient to represent acceptance conditions as propositional formulas. For this reason we will frequently use a logical representation of ADFs (S, L, C) where C is a collection $\{\varphi_s\}_{s \in S}$ is called acceptance condition of s.
>
> In many cases it is convenient to represent acceptance conditions as propositional formulas. For this reason we will frequently use a lgoical representation of ADFs (S, L, C) where C is a collection $\{\varphi_s\}_{s \in S}$ of propositional formulas.

Moreover, unless specified differently we will tacitly assume that the acceptance formulas specify the parents a node depends on implicitly. It is then not necessary to give the links in the graph explicitly. We thus can represent an ADF D as a tuple (S, C) where S and C are as above and L is implicitly given as $(ab) \in L$ iff a appears in φ_b.[11]

The different semantics of ADFs over statements S are based on the notion of a model. A two-valued interpretation v — a mapping from statements to the truth balues true and false — is a *two-valued model* (*model*, if clear from the context) of an ADF (S, C) whenever for all statements $s \in S$ we have $v(s) = v(\varphi_s)$, that is v maps exactly those statements to true whose acceptance conditions are satisfied under v. Our analysis in this paper will be based on a straightforward generalization of two-valued interpretations for ADFs to Kleene's strong three-valued logic [19].[12] A three-valued interpretation is a mapping $v : S \rightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ that assigns one of he truth values true ($\mathbf{t}$), false $\mathbf{f}$) or unknown ($\mathbf{u}$) to each statement. Interpretations can easily be extended to assign truth values to propositional formulas over the statements: negation switches $\mathbf{t}$ and $\mathbf{f}$, and leaves $\mathbf{u}$ unchanged; a conjunction is $\mathbf{t}$ if both conjuncts are $\mathbf{t}$, it is $\mathbf{f}$ if some conjunct is $\mathbf{f}$ and it is $\mathbf{u}$ otherwise; disjunction is dual. It is also straightforward to generalize the notion of a model: a three=valued interpretation is a model whenever for all statements $s \in S$ we have $v(s) \neq \mathbf{u}$ implies $v(s) = v(\varphi_s)$.

The three truth values are partially ordered by $\leq_i$ according to their information content: we have $Bu <_i \mathbf{t}$ and $\mathbf{u} <_i \mathbf{f}$ and no other pair in $<_i$, which intuitively means that the classical truth valuecontain more information than the truth value unknown. The pair $(\{\mathbf{t}, \mathbf{f}, \mathbf{u}\} \leq_i)$ forms a complete meet-semilattice[13] with the meet operation $\sqcap$. This meet can be read as *consensus* and assigns $\mathbf{t} \sqcap \mathbf{t} = \mathbf{t}, \mathbf{f} \sqcap \mathbf{f} = \mathbf{f}$, and returns $\mathbf{u}$ otherwise.

End quote from [35].

Let us conclude this Appendix and quote here from the Brewka–Woltran paper. They are aware that they are really doing logic programming but, I assume for their own reasons, they still present their paper as an argumentation paper. See also [9, 10, 12]. We present three quotations from [2].

Begin quote 1:
Vocabulary for the quotation below: s is a node. $Par(s)$ are its attackers, it is Y_s in our notation. C_s is our $I(s)$.

[11] When presenting examples we will use a notation where acceptance conditions are written in square brackets behind nodes, e.g. $c[\neg(a \wedge b)]$ denotes a node a which is jointly attacked by nodes a and b, that is, each attacker alone is insufficient to defeat c.

[12] A comparable treatment for AFs was given by the labellings of [17]. We use standard notation and terminology from mathematical logic.

[13] A complete meet-semilattice is such that every non-empty finite subset has a greatest lower bound, the meet; and every non-empty directed subset has a least upper bound. A subset is directed if any two of its elements have an upper bound in the set.

Definition 1. An abstract dialectical framework is a tuple $D = (S, L, C)$ where

- S is a set of statements (positions, nodes),
- $L \subseteq S \times S$ is a set of links,
- $C = \{C_s\}_{s \in S}$ is a set of total functions $C_s : 2^{par(s)} \to \{\text{in, out}\}$, one for each statement s. C_s is called acceptance condition of s.

Begin quote 2:
Definition 2. Let $D = (S, L, C)$ be an ADF. $M \subseteq S$ is called conflict-free (in D) if for all $s \in M$ we have $C_s(M \cap par(s)) = \text{in}$. Moreover, $M \subseteq S$ is a model of D if M is conflict-free and for each $sinS, C_s(M \cap par(s)) = \text{in}$ implies $s \in M$.

In other words, $M \subseteq S$ is a model of $D = (S, L, C)$ if for all $s \in S$ we have $s \in M$ iff $C_s(M \cap par(s)) = \text{in}$.

We say M is a minimal model if there is no model M which is a proper subset of M.

Begin quote 3:
It is not difficult to verify that, when the acceptance condition of each node s is represented as a propositional formula $F(s)$, a model is just a propositional model of the set of formulas

$$\{s \equiv F(s) | s \in S\}.$$

End quotes from [2].

Remark D.1 *To further highlight the fact that the idea of Instantiation is different from and is orthogonal to the ADF idea, let us define the notion of abstract Instantiated Dialectical Frameworks.*

An abstract dialectical framework is a tuple $DI = (S, L, C, I)$ where

- *S is a set of statements (positions, nodes),*
- *$L \subseteq S \times S$ is a set of links,*
- *$C = \{C_s\}, s \in S$ is a set of total functions $C_s : 2^{par(s)} \to \{in, out\}$, one for each statement s. C_s is called acceptance condition of s.*
- *For each $s \in S, I(s)$ is a wff of classical propositional logic. We consider I(s) as a function from $2^{par(s)} to \{in, out\}$.*

Following quote 3 above , we understand by a model of the system any solution to the equations for $s \in S$,

$$I(s) = C_s(y/I(y) | y \text{ a parent of } s).$$

This system of equations is not guaranteed a solution is the real interval $[0, 1]$, while the system of Quote 3, does always have a solution in $[0, 1]$, though not always in $\{0, 1/2, 1\}$.

E

E.1 Comparing abstract instantiation with the ASPIC approach

When we talk in this paper about "instantiation in argumentation", we must compare what we are doing with the well known school of "instantiated argumentation networks" and the ASPIC movement [13, 31]. What is the connection between what this paper is doing and ASPIC? The answer is that they are similar but different. We first explain the difference in principle and then give examples.

1. The abstract instantiation of the current paper. In this paper we start with an abstract argumentation frame (S, R) and in parallel with a closed logical theory Δ.[14] It is important to note that Δ need not be a defeasible theory. In fact our main case studies are monotonic logics; classical propositional logic, monadic predicate logic and modal logic S5. We can also use a defeasible theory if we want. We substitute wffs of Δ into S via a function $I : S \mapsto \Delta$. In this set up the network (S, R) and the attack structure R of the argumenation frame is primary and it is retained and is influenced by the substitution function I. Our main task is to give the system (S, R, I) proper meaning. This substitution may cause problems and we need to find a theoretical remedy.

For the purpose of comparing with ASPIC and with other instantiation approaches, such as [8, 28, 29, 30], let us offer another way of looking at the same problem. It is to say that we have a basic consistent closed theory Δ and in parallel we have an abstract schema (S, R) of attack relation. By substituting, using I, formulas from Δ for the elements of S, we form an indexed collection of formulas $\Gamma = \{I(x) | x \in S\}$ of wffs from Δ containing an abstract unspecified conflict, as recorded by R. If this abstract conflict could be expressed in the language of Δ as a theory Δ_R, then $\Delta \cup \Delta_R$ would have been a defeasible theory or a monotonic consistent theory or a monotonic inconsistent theory, all depending on the underlying logic. We ask the question: is it possible to retain consistency of Γ and yet satisfy the constraints R?

To motivate this point of view of constraints, and to be able to have a working case study to use in comparing with ASPIC, let us give an example.

Example E.1 *Let Δ be a theory governing a birthday party. Let S be a set of people and let $I(x)$ for $x \in S$ be "x is invited to the party". Suppose everything is consistent together, namely $\Delta \cup \{I(x) | x \in S\}$ is consistent. We can have the party as required by Δ and invite all the people of S. Bear in mind that Δ may contain requirements which may affect the people invited.*

We now add the constraints R of the form xRy meaning "If you invite x you cannot invite y".

OK, our problem now is, whom do we invite and still satisfy the constraints? In other words, we are looking for an extension (in fact a maximal preferred extension) of (S, R, I).

[14] By a closed theory we mean a theory containing whatever it proves. This can also be defined not only for monotonic theories but also for non-monotonic theories which satisfy the restricted monotonicity rule, namely:

$$\Delta \mid\!\sim \Phi \text{ and } \Delta + \Phi \mid\!\sim \Psi \implies \Delta \mid\!\sim \Psi.$$

2. Instantiated argumentation: ASPIC. In comparison ASPIC does something different. It will look at all the arguments (proofs) generated by a defeasible theory Δ. Since Δ is defeasible, it might defeasibly prove both an x and its negation $\neg x$. This can be viewed as giving rise to an attack relation (several possible notions of attack relations) among the set of all proofs from the theory Δ. We need to resolve the conflict. ASPIC will use argumentation theory and *construct* an instantiated network (S, R) accordingly. Then ASPIC would apply the machinery of finding extensions and *expect* the resulting wffs of the extensions to be consistent in the logic of Δ.

When this does not happen, ASPIC offers postulates to *force* Δ to behave as to ensure that the resulting extensions are consistent.

Thus the ASPIC process has three stages:

1. **The Input**
   Input a defeasible theory Δ.

2. **The System**
   Construct an argumentation network (S, R) from Δ. Call the process of construction the 'ASPIC construction". This is ASPIC's way of doing it.

3. **The Output**
   Output complete extension of the network (S, R) constructed in 2 and expect these complete extensions to be consistent in the logic of Δ.

When the output 3 does not meet expectations, ASPIC restricts the input by putting postulates on Δ. ASPIC does not try and improve or change the construction in point 2. (We shall offer a different way of constructing a network in Appendix E2).

3. Summary comparison. As you can see, although our paper and ASPIC deal with similar (lego pieces) components, they do not do the same things. Several examples would be helpful.

Example E.2 *We look at Example 6 from the Caminada and Amgoud paper [16].*

A defeasible theory Δ is given with strict rules $\mathcal{S}$ and defeasible rules $\mathcal{D}$ where

$$\mathcal{S} = \{a, d, g, b \wedge c \wedge e \wedge f \rightarrow \neg g\}$$
$$\mathcal{D} = \{a \Rightarrow b, b \Rightarrow c, e \Rightarrow f, d \Rightarrow e\}$$

and where "$\rightarrow$" is strict implication and $\Rightarrow$ is defeasible implication. There are two modus ponens rules, one for each implication.

Following the ASPIC instantiation idea, we look at all possible proofs of atoms from Δ. These are (notation: $[\ldots]$):

$\Pi 1$. $[a]$

$\Pi 2$. $[d]$

$\Pi 3$. $[g]$

$\Pi 4$. $[a, a \Rightarrow b]$, *(proving b)*

$\Pi 5.$ $[d, d \Rightarrow e]$, *(proving e)*

$\Pi 6.$ $[a, a \Rightarrow b, b \Rightarrow c]$, *(proving c)*

$\Pi 7.$ $[d, d \Rightarrow e, e \Rightarrow f]$, *(proving f)*

$\Pi 8.$ $[4, 5, 6, 7, b \wedge c \wedge e \wedge f \rightarrow \neg g]$, *(proving $\neg g$)*

The paper [16] uses the proofs $\Pi 1$–$\Pi 7$ *to construct an argumentation network*

$$S_\Delta = \{\Pi 1, \Pi 2, \Pi 3, \Pi 4, \Pi 5, \Pi 6, \Pi 7\}$$
$$R_\Delta = \{\Pi R \Pi' \textit{ if } \Pi \textit{ proves } x \textit{ and } \Pi' \textit{ proves } \neg x\}.$$

The way [16] defines S_Δ it turns out that the proof $\Pi 8$ *cannot be used and is not a member of S_Δ. This is because they string the data without using conjunctions, using only "linear" sequences of the form $x_1, x_1 \rightsquigarrow x_2, x_2 \rightsquigarrow x_3, \ldots$, where "$\rightsquigarrow$" is either "$\rightarrow$" or "$\Rightarrow$" and where each x_i is atomic. This way of constructing (S_Δ, R_Δ) causes problems.*

If we collect the atoms proved in the proofs of S_Δ and which are in the ground extension, we get $\Gamma = \{a, d, g, b, e, c, f\}$. Using the logic of Δ and the rule $b \wedge c \wedge e \wedge f \rightarrow \neg g$, we get that Γ is contradictory. Thus taking extensions may result in inconsistent sets.

So [16] offers postulates on theories Δ, hoping to ensure there will be no problems.

We are not here to evaluate [16] or the ASPIC approach. We are just comparing their idea of "instantiation" with ours.

- *ASPIC starts with Δ and constructs (S_Δ, R_Δ).*
- *We start with (S, R) and substitute proofs from Δ.*

We could end up with similar difficulties. Suppose we start with

$$S = \{x_1, \ldots, x_8\}, R = \emptyset$$

and $I(x_i) = \Pi i$. We end up with

$$S_I = \{\Pi_1, \ldots, \Pi_8\}.$$

The question is, in our methodology, how do we define R_I? Since we allow for joint attacks, we can express Π_8. Let us play the game the ASPIC way. We are given Δ, let us ask, since we have powerful machinery in this paper, if we wanted to construct (S_Δ, R_Δ), how would we have done it?[15]

Our answer is that we would construct a 2-state network with joint attacks. Figure 67 illustrates the network we get.

We need two notions of attack, strict attack $x \twoheadrightarrow y$ (same as $a \rightarrow \neg y$ and defeasible attack $x \Rrightarrow y$ (same as $x \Rrightarrow \neg y$). We have $\neg\neg x = x$ and $\neg x \twoheadleftarrow\twoheadrightarrow x$.

We need joint attacks as shown in figure 67. We also need to use $\top$, $\top$ is truth, *helping us describe the strict facts.*

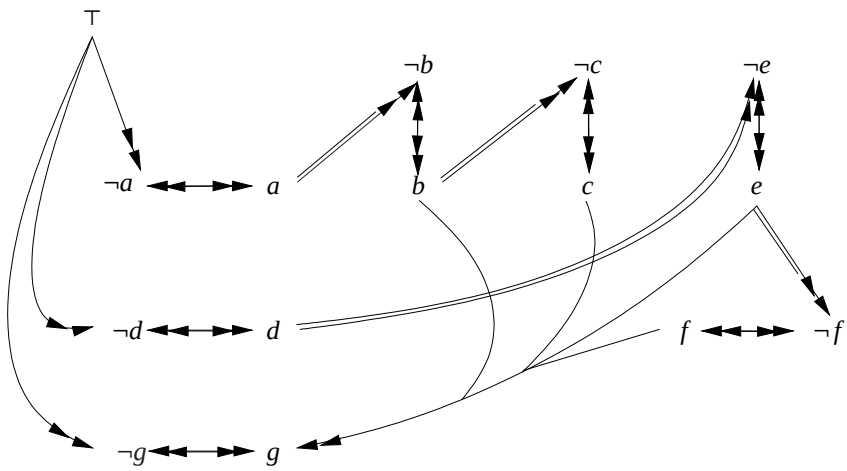
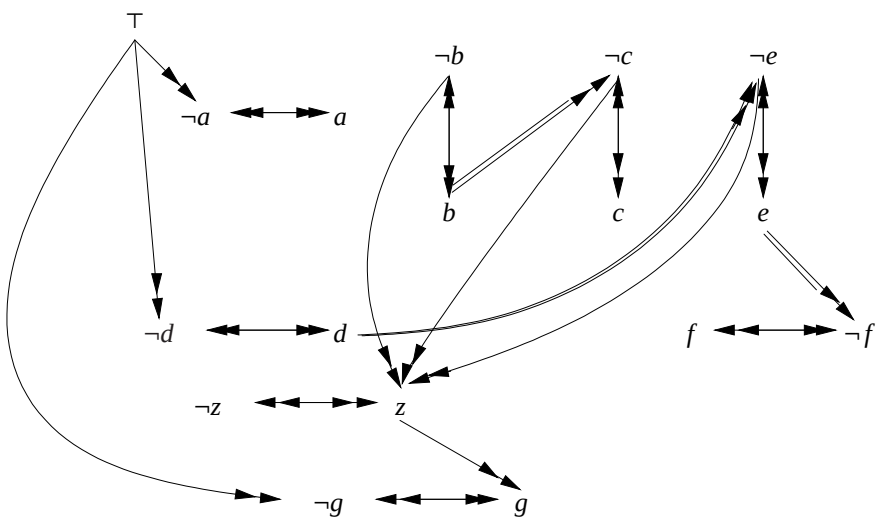


Figure 67:



$b \wedge c \wedge e \wedge f \twoheadrightarrow g$ is replaced by $\neg b \twoheadrightarrow z$ and $\neg c \twoheadrightarrow z$ and $\neg e \twoheadrightarrow z$ and $\neg f \twoheadrightarrow z$ and $z \twoheadrightarrow g$.

Figure 68:

Note that in view of Remark C.6, Figure 67 is equivalent to Figure 68. This figure contains node to node attacks only.

We need to say how we compute extensions for the argumentation network of Figure 67 arising from the defeasible theory Δ. Being a defeasible theory we should be able to resolve the conflict of the atom g both supported and attacked by its neighbours. Intuitively g has a direct strict proof from $\top$, but $\neg g$ is derived defeasibly using 3 separate defeasible rules, two of which are chained. So $\neg g$ has lower preference "value" than g and so g is accepted. We need to define the priority values on the atoms $x, \neg x, y, \neg y$, etc., and use these in calculating extensions. Our network has two attack relations and so the priority values and the extensions must be defined geometrically in terms of the graphs. This is done in Appendix E2.

Example E.3 *Let us look at another example from [16]. This is example 4 from their paper. The vocabulary is:*

wr*: John wears a ring*
m*: John is married*
hw*: John has a wife*
b*: John is a bachelor*
go*: John often goes out until late with friends.*

The arguments are obtained from a database with strict rules $\mathcal{S}$ and defeasible rules $\mathcal{D}$.

$$\mathcal{S} = \{\top \to wr, \top \to go, b \to \neg hw, m \to hw\}$$
$$\mathcal{D} = \{wr \Rightarrow m, go \Rightarrow b\}.$$

Caminada and Amgoud form the following arguments from the data $(\mathcal{S}, \mathcal{D})$.

$$A_1 : [\top \to wr]$$
$$A_2 : [\top \to go]$$
$$A_3 : [A_1 \Rightarrow m]$$
$$A_4 : [A_2 \Rightarrow b]$$
$$A_5 : [A_3 \to hw]$$
$$A_6 : [A_4 \to \neg hw]$$

We form (in our notation) the following network of Figure 69.

The correspondence in this case between the arguments of Caminada and Amgoud and the arguments in Figure 69 is to paths in this figure as follows:

$$A_1 : (\top \to wr)$$
$$A_2 : (\top \to go)$$
$$A_3 : (\top \to wf \Rightarrow m)$$
$$A_4 : (\top \to go \Rightarrow b)$$
$$A_5 : (\top \to wr \Rightarrow m \to hw)$$
$$A_6 : (\top \to go \Rightarrow b \to \neg hw)$$

[15] Recall the discussion in Section 1.2. ASPIC restricts the input, i.e. using (r1). We are going to change the system, the way we construct the network, i.e. we are using (r2).

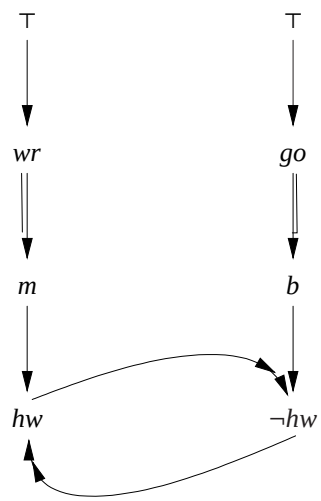


Figure 69: Representation of Caminada and Amgoud example

In the above set of arguments, only A_5 and A_6 attack each other and so we get the extension $\{A_1, A_2, A_3, A_4\}$. The output *of this extension are the argument heads, namely $\{wr, go, m, b\}$.*

Caminada and Amgoud's approach corresponds to our considering the network of Figure 69, with attack relation $\twoheadrightarrow$, and we thus get the extension $\{wr, go, m, b\}$. Caminada and Amgoud proceed to close this extenson under the strict rules and they get inconsistency. They consider this a problem and offer to remedy the problem by systematically adding, with every strict rule $x \rightarrow y$, its contrapositive rule $\neg y \rightarrow \neg x$. This allows them essentially to also have that m attacks b and b attacks m, and so the extension becomes only $\{wr, go\}$, which is consistenly closed under the strict rules.

We translate Figure 69 into Figure 70, and so have no problems with it..

E.2 Defining complete extensions for two-state two-attacks abstract argumentation networks

Definition E.4 *A 2-state bipolar 2-attack network has the form $(S \cup S^{\neg} \cup \{\top\}, R_1, R_2)$ where S is a set of atomic letters, $S^{\neg} = \{\neg x | x \in S\}$, $\top \notin S$ and R_1 and R_2 are subsets of $(S \cup S^{\neg} \cup \{\top\})^2$. We write $x \twoheadrightarrow y$ for $(x, y) \in R_1$ and $x \Rrightarrow y$ for $(x, y) \in R_2$.*

We consider the elements of $S^{\neg}$ as negations of the elements of S. $\top$ is truth.

The following holds (compare with Definition 1.2):

- $\neg\exists x(x \twoheadrightarrow \top$ *or* $x \Rrightarrow \top)$
- $\forall x(x \twoheadrightarrow \neg x$ *and* $\neg x \twoheadrightarrow x)$.

R_1 is called the strict attacks and R_2 is called the defeasible attacks.

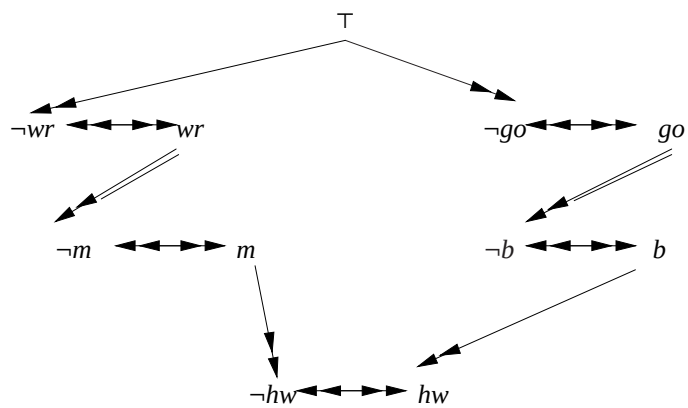


Figure 70:

Example E.5 *Consider the network of Figure 68. This is a 2-state two-attack bipolar network. We need to define a process for finding extensions for it. The difficult part in the definition of such a notion is to deal with cases where the attacks "$\twoheadrightarrow$" and "$\Rightarrow\!\!\!\!\Rightarrow$" disagree. By calling "$\twoheadrightarrow$" strict and "$\Rightarrow\!\!\!\!\Rightarrow$" defeasible we are giving "$\twoheadrightarrow$" priority over "$\Rightarrow\!\!\!\!\Rightarrow$". But we may have cases where "$\Rightarrow\!\!\!\!\Rightarrow$" attacks both x and $\neg x$. We might give priority to either x or $\neg x$ depending on the geometrical location of $x, \neg x$ relative to their attacking "$\Rightarrow\!\!\!\!\Rightarrow$" ancestors.*

Let us see how to calculate the ground extension by propagating the attacks from $\top$. We record a defeasible index $\mathbf{D}(x)$ for every node x, by counting the maximal number of "$\Rightarrow$" participating in the chain of attacks leading up to it.

The following is the progression, step by step. 1= "in", 0= "out", $\frac{1}{2}$="undecided".

1. $\top = 1$ — $\mathbf{D}(\top = 0)$
2. $\neg g = 0, g = 1,$ $\mathbf{D}(g) = \mathbf{D}(\neg g) = 0$ — *from 1*
3. $\neg d = 0, d = 1,$ $\mathbf{D}(d) = \mathbf{D}(\neg d) = 0$ — *from 1*
4. $\neg a = 0, a = 1,$ $\mathbf{D}(a) = \mathbf{D}(\neg a) = 0.$ — *from 1*
5. $\neg e = 0, e = 1,$ $\mathbf{D}(e) = \mathbf{D}(\neg e) = 1$ — *from 3*
6. $\neg b = 0, b = 1,$ $\mathbf{D}(b) = \mathbf{D}(\neg b) = 1$ — *from 4*
7. $\neg c = 0 c = 1,$ $\mathbf{D}(c) = \mathbf{D}(\neg c) = 2$ — *from 6*

8. $\neg f = 0. f = 1$ *from 5*
   $\mathbf{D}(f) = \mathbf{D}(\neg f) = 2$

9. $\neg g = 1, g = 0$ *from 5, 6, 7, 8*
   $\mathbf{D}(g) = \mathbf{D}(\neg g) = 2.$

We see that 2 contradicts 9. Since the **D** *index of 2 is lower than that of 9, it wins.*

Therefore our ground extension for the network of Figure 67 is

$$\{\top, g, d, a, e, b, c, f\}.$$

The reader can see that by associating with a theory Δ the network of Figure 67 we have none of the problem mentioned by the ASPIC group [16].

Furthermore, if one does not want to deal with joint attacks (see [3] and see [18] for arguments in favour of joint attacks), one can use additional auxiliary points and eliminate them as in Figure 68.

Example E.6 *Let us calculate the ground extension of Figure 70 in steps:*

1. $\top = 1$. $\mathbf{D}(\top) = 0$
2. $\neg go = 0, go = 1, \mathbf{D}(\neg go) = \mathbf{D}(go) = 0$, *from 1.*
3. $\neg wr = 0, wr = 1, \mathbf{D}(\neg wr) = \mathbf{D}(wr) = 0$, *from 1.*
4. $\neg m = 0, m = 1, \mathbf{D}(\neg m) = \mathbf{D}(m) = 1$, *from 3.*
5. $\neg b = 0, b = 1, \mathbf{D}(\neg b) = \mathbf{D}(b) = 1$, *from 2*
6. $\neg hw = 0$. $\mathbf{D}(\neg hw) = 1$, *from 4.*
7. $hw = 0, \mathbf{D}(hw) = 1$, *from 5.*

If we insist on the sum of the values of any $\{w, \neg w\}$ to be 1 we need to give hw and $\neg hw$ values $\frac{1}{2}$.

Example E.7 *We want to discuss options for defining the index $\mathbf{D}(x)$ for index x. Consider the network in Figure 71.*

The node c is supported by several chains. For example

1. $\top \Rightarrow a \Rightarrow c$
2. $\top \Rightarrow a \Rightarrow b \Rightarrow a \Rightarrow c$
3. $\top \Rightarrow d \Rightarrow c$
4. $\top \Rightarrow e \Rightarrow c$
5. $\top \Rightarrow c$

We can also loop and get

$$(6, k)\ \top \Rightarrow a \Rightarrow b \Rightarrow a, \ldots,\ \textit{loop k times}\ \Rightarrow a \Rightarrow b \Rightarrow a \Rightarrow c.$$

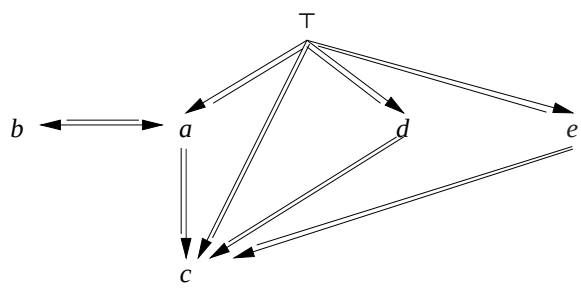


Figure 71:

length	number of paths
2	1
3	3
5	1
7	1
9	1
11	1
13	1
15	1

Figure 72:

Questions. *What index do we give to c?*

The first question is what to do with (6,k). We have an infinite number of paths leading from $\top$ *to c. The second question is that we can get to c using several parallel paths through* a, d *and e. Does this increase its priority over* f*?*

The thrid question is how do we record the totality of inidices/paths which characterise c?

We think it is reasonable to do the following:

1. *Limit the paths with loops to n number of repetitions, where n is the number of elements in the network. Recall that we are dealing with finite networks. Thus* $(6, k)$ *becomes* $(6, 1), \ldots, (6, 6)$.

2. *Record for each x the number of different paths of each length. Thus the index of node c becomes the table in Figure 72.*

We leave it to the defeasible logical theory to decide which x with table $\mathbf{D}(x)$ *is preferable to which y with table* $\mathbf{D}(y)$*. For example we can take the number of shortest paths to be the index.*

Definition E.8 *Let* $(S \cup S^{\neg} \cup \{\top\}, R_1, R_2\}$ *be a finite 2-state 2-attack bipolar network. Let k be the number of elements in the network.*

1. *By a legitimate path from x to* $\top$ *we mean a sequence*

$$\Pi_x = (x_1 = x, x_2, \ldots, x_n = \top)$$

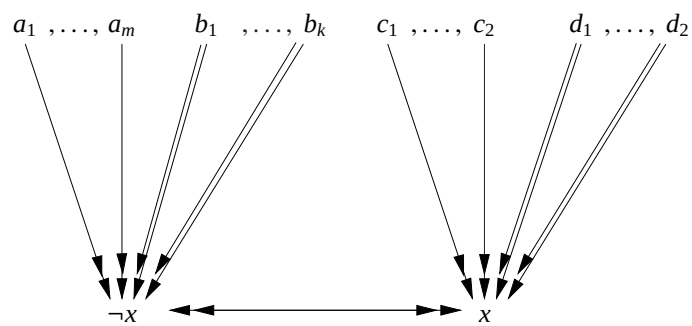


Figure 73:

such that for each $1 < i \leq n$ *we have* $x_i R x_{i-1}$*, where* R *is either* $R_1 = \twoheadrightarrow$ *or* $R_2 = \Rrightarrow$ *and such that no node* y *appears in* Π_x *more than* k *times.*

2. *The defeasible value of* Π_x *(denoted by* $\mathbf{D}(\Pi_x)$*) is the number of points* $x_i, 1 < i \leq n$ *such that* $x_i \Rrightarrow x_{i-1}$*.*

3. *The defeasible value of* x *denoted by* $\mathbf{D}(x)$ *is the pair* $(\mathbf{D}_1(x), \mathbf{D}_2(x))$ *where* $\mathbf{D}_1(x) = \min\{\mathbf{D}(\Pi_x)\}$ *and* $\mathbf{D}_2(x) =$ *the number of paths* Π_x *such that* $\mathbf{D}(\Pi_x) = \mathbf{D}_1(x)$*.*

4. *Order the nodes* x *according to the lexicographic ordering of* $\mathbf{D}(x)$*. We get three possibilities for two nodes* x, y*.*

   (a) $\mathbf{D}_1(x) < \mathbf{D}_1(y)$

   (b) $\mathbf{D}_1(x) = \mathbf{D}_1(y)$ *and* $\mathbf{D}_2(x) > \mathbf{D}_2(y)$

   (c) $\mathbf{D}_1(x) = \mathbf{D}_1(y)$ *and* $\mathbf{D}_2(x) = \mathbf{D}_(y)$*.*

   Possibilities (a) and (b) are considered $x > y$ *(*x *is stronger than* y*). Possibility (c) is considered* $x \approx y$ *(*x *is indifferent to* y*).*

Definition E.9 *Let* $(S \cup S^{\neg} \cup \{\top\}, \twoheadrightarrow, \Rrightarrow)$ *be a finite 2-state, 2-attack bipolar network. Assume that each node* x *already has an index* $\mathbf{D}(x)$ *defined and that a priority ordering* $x > y$ *and* $x \approx y$ *have been defined. We define the notion of a Caminada-Gabbay labelling function*

$$\lambda : S \cup S^{\neg} \cup \{\top\} \mapsto \{0, \tfrac{1}{2}, 1\}$$

being a legitimate complete extension for the network.

λ is defined relative to the relations $>$ *and* $\approx$*.*

λ must satisfy the following

1. $\lambda(\top) = 1$.

2. *Each pair* $\{x, \neg x\}$ *in the network is part of the following geometrical constellation of Figure 73.*

   There may be no a_i *and/or no* b_j *and/or no* c_i *and/or no* d_j*.*

We assume that the λ function is known for {as, bs, cs, ds} and we indicate by case analysis what the values $\lambda(\neg x)$ and $\lambda(x)$ should be.

3. $\lambda(x) + \lambda(\neg x) = 1$.

4. *It is not the case that for some i and some j,*

$$\lambda(a_i) = \lambda(c_j) = 1$$

(If this happens then λ is not legitimate.)

For networks coming from consistent defeasible theories Δ, this will not happen because it means that using strict rules we have $a_i \vdash \neg x$ and $c_j \vdash x$ and both $\Delta \vdash a_i$ and $\Delta \vdash c_j$.

5. *If for some i, $\lambda(a_i) = 1$ and for all j, $\lambda(c_j) < 1$ then $\lambda(\neg x) = 0$ and $\lambda(x) = 1$.*

6. *If for some i, $\lambda(c_i) = 1$ and for all j, $\lambda(a_j) < 1$ then $\lambda(\neg x) = 1$ and $\lambda(x) = 0$.*

7. *Assume that the values $\lambda(a_j), \lambda(c_i)$ are < 1 for all a_j and c_i. If for at least one of $\{a_j, c_i\}$, λ gives value $\frac{1}{2}$ then $\lambda(x) = \lambda(\neg x) = \frac{1}{2}$.*

8. *Assume λ gives value 0 to all $\{a_i, c_j\}$, and assume that $x > \neg x$, then*

   (a) *If for some d_j, $\lambda(d_j) = 1$ then $\lambda(x) = 0$ and $\lambda(\neg x) = 1$.*

   (b) *If for all d_j, $\lambda(d_j) < 1$ and for some d_j $\lambda(d_j) = \frac{1}{2}$ then $\lambda(x) = \lambda(\neg x) = \frac{1}{2}$.*

9. *Assume for all {as, cs, ds} λ gives value 0 and $x > \neg x$ then*

   (a) *If for some b_j, $\lambda(b_j) = 1$ then $\lambda(\neg x) = 0$ and $\lambda(x) = 1$.*

   (b) *If for all b_j, $\lambda(b_j) < 1$ and for some b_j, $\lambda(b_j) = \frac{1}{2}$ then $\lambda(x) = \lambda(\neg x) = \frac{1}{2}$.*

   (c) *If for all j, $\lambda(b_j) = 0$ (i.e. the case is that none of $\neg x, x$ is attacked in any way) then $\lambda(x) = \lambda(\neg x) = \frac{1}{2}$.*

10. *The mirror case of (7)–(8) for $\neg x > x$. Take the mirror case analysis.*

11. *If $x \approx \neg x$ and all $\lambda(as) = \lambda(cs) = 0$ and $\lambda(b_j) = 1$ for some j and all $\lambda(ds) < 1$ then $\lambda(\neg x) = 0$ and $\lambda(x) = 1$.*

12. *If all $\lambda(as) = \lambda(cs) = 0$ and all $\lambda(bs) < 1$ and some $\lambda(ds) = 1$ then $\lambda(\neg x) = 1$ and $\lambda(x) = 0$.*

13. *If all $\lambda(as) = \lambda(cs) = 0$ and either for some b_j and d_i $\lambda(b_j) = \lambda(d_i) = 1$ or all $\lambda(bs), \lambda(ds) < 1$ then $\lambda(\neg x) = \lambda(x) = \frac{1}{2}$.*

Definition E.10 *Consider a rule of the form $\alpha = [\bigwedge_i x_i \rightsquigarrow z]$ where "$\rightsquigarrow$" is either "$\twoheadrightarrow$" or "$\Rrightarrow$", and x_i and z are literals of the form either b or $\neg b$, with b atomic. We agree that "$\neg\neg b$" is "b".*

We translate α into an attack formation Δ_α in the language of 2-state 2-attack networks as in Figure 41. We use the auxiliary points $S_\alpha = \{y_1(\alpha), \ldots, y_n(\alpha), y(\alpha)\}$

$$\Delta_\alpha = \{x_i \rightsquigarrow\!\!\!\rightarrow \neg y_k(\alpha) | i = 1, \ldots, n\} \cup \{y_i(\alpha) \rightsquigarrow\!\!\!\rightarrow \neg y(\alpha), y(\alpha) \rightsquigarrow\!\!\!\rightarrow \neg z\}.$$

Where "$\rightsquigarrow\!\!\!\!\rightarrow$" is "$\twoheadrightarrow$" if "$\rightsquigarrow$" is "$\rightarrow$" and "$\rightsquigarrow\!\!\!\!\rightarrow$" is "$\Rrightarrow$" if "$\rightsquigarrow$" is "$\Rightarrow$".

The auxiliary points of S_α, S_β are all pairwise disjoint for α different from β and disjoint from the literals of Δ.

Definition E.11 *Let Δ be a defeasible theory based on the set of atoms S, containing the literals $S \cup S^\neg$ with $S^\neg = \{\neg x | x \in S\}$, with $\bot, \top \notin S \cup S^\neg$.*

Let $\mathcal{D}$ be the set of defeasible rules and $\mathcal{S}$ be the set of strict rules. Assume the language of Δ has strict implication $\rightarrow$ and defeasible implication $\Rightarrow$. The rules of Δ have the form

$$\bigwedge_i x_i \rightarrow y$$

or

$$\bigwedge_j y_j \Rightarrow z$$

where $\{x_i, y_i, y, z\}$ are literals, i.e. have the form b or $\neg b$, b atomic letter.

We allow for rules of the form $\rightarrow y$ or $\Rightarrow z$ meaning y or z are assumptions.

We define the associated network $N(\Delta)$ for Δ as follows:

1. *The set of nodes of $N(\Delta)$ is $S \cup S^\neg \cup \{\top\} \cup \bigcup_\alpha (S_\alpha \cup S_\alpha^\neg)$ where α runs over all rules of the form $\bigwedge_{i=1}^k x_i \rightsquigarrow z$ with $k \geq 2$. (I.e. joint rules) and "$\rightsquigarrow$" is either "$\rightarrow$" or "$\Rightarrow$".*

   The attack relation of $N(\Delta)$ is as follows:

   $$\{b \twoheleftarrow \neg b | b \text{ atom of } N(\Delta\} \cup \{\top \twoheadrightarrow \neg b | c \twoheadrightarrow b \text{ in } \Delta\} \cup \{\top \Rrightarrow \neg b | c \Rightarrow b \text{ in } \Delta\} \cup \bigcup_\alpha \Delta_\alpha \cup \{x \rightsquigarrow\!\!\!\!\rightarrow \neg y | x \rightsquigarrow y \text{ in } \Delta\}$$

   where α runs over all joint rules α in Δ.

   Note that $N(\Delta)$ satisfies that for every $b \neq \top$ either b or $\neg b$ is attacked (using $\rightsquigarrow\!\!\!\!\rightarrow$) by some node.

F Discussion of papers of Arieli and Strasser [28, 29] and the book of Besnard and Hunter [30]

Our purpose here is to compare our work with that of Besnard and Hunter and in parallel, with that of Arieli and Strasser. We first want to make two comments:

1. To set the scene for the comarison we need to start with my 1999 paper [32], which contains the relevant machinery.

2. Whatever criticism I have here of Arieli and Strasser [28, 29], it must be borne in mind that these papers are preliminary conference papers and not definitive versions, like, e.g. Besnard and Hunter [30].

So let us start. My paper [32] and the later chapter 7 of our mootgraph [33] dealt with what I called *compromise revision* of databases. We explain by example.

Example F.1 *This example continues Example E.1, with a view of illustrating the idea of compromise revision.*

Let Δ be a theory governing a birthday party. Add to Δ the following additional statements (t1)–(t5)

(t1) a = *invite Agnes*

(t2) b = *invite Bertha*

(t3) $a \to \neg b$

(t4) $b \to \neg a$

(t5) $a \wedge b \to$ *invite Caterina (let c = invite Caterina).*

Then our database is Γ:

$$\Gamma = \Delta \cup \{a, b, a \to \neg b, b \to \neg a, a \wedge b \to c\}.$$

Γ is not consistent, but each of its items makes initial sense. Agnes and Bertha may be old aunties who do not talk to each other because of something that happened 30 years ago. Caterina may be an old friend of each one of them whose presence might "mitigate" the friction. After some deliberation a decision was made not to risk inviting these two warring aunties. This means that we regard Γ as an inconsistent theory in need of belief revision. The obvious revision is to delete either (t1) or (t2), i.e. not invite one of the aunties. If we do that then we need not invite Caterina (i.e. c would not be provable).

The compromise revision problem is the following:

- *Given an inconsistent Γ and $X \in \Gamma$ such that $\Gamma - X$ is consistent, and given a Z such that $\Gamma \vdash Z$ but $\Gamma - X \nvdash Z$, then compromise revision would like to include Z in the revised theory in case the revised theory is $\Gamma - X$.*

The problem is that if Γ is inconsistent, then Γ can prove everything, including

$$V = \textit{invite Vladimir Putin}$$

So we need to be careful and "control" what Γ exactly proves. For that we use the discipline of labelled deductive systems [34]. We label every step in the syntactic proof of any wff Z.

The LDS rule of Modus Ponens is

$$\frac{\beta : A; \gamma : A \to B}{(\gamma, \beta) : B}$$

Thus we have that Γ proves

$$(t_1, t_2, t_5) : c$$

If we can prove V, it will be by some label α, $\alpha : V$ which can be recognised.

So to prove V by virtue of Γ being inconsistent, we will need to first prove some $x \wedge \neg x$ and then use the axiom $x \wedge \neg x \to V$. This will all be recorded in α and we can recognise it and not include V in the revised theory, i.e. invite Vladimir Putin. By comparison, if for example Putin is a relative of Agnes and Δ says something about relatives, we may have a more direct labelled proof of V.

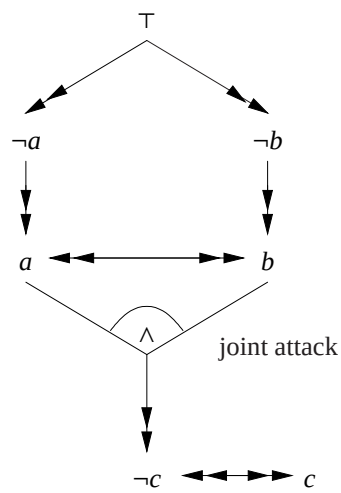


Figure 74:

The background considerations in Example F.1 show that we have precise LDS machinery to trace proofs. Thus the Besnard and Hunter notion of argument of the form (Δ, A) where Δ is a minimal theory such that $\Delta \vdash A$, can be refined to be $t_\Delta : A$ where t is a label showing how A is proved from Δ. There may be several such proofs in which case there would be several such labels. So given an inconsistent theory Γ, we can look at all labelled proofs $t_i : A_i$ from Γ and define an attack relation among them in a much refined way, taking into account exactly how each formula is proved.

Arieli and Strasser use a Gentzen formulation of the logic and use progressions of Gentzen sequents as their attacking elements. I have reservations about the very idea of using Gentzen systems as the machinery for attack in the context of argumentation networks. I think that Arieli and Strasser's impressive system is more at home with meta-level theories of belief revision, rather than abstract argumentation. However, this is not the place to to discuss and evaluate their system, especially since, the current publication is at a conference and we yet to wait for the Journal expanded version. It is enough to say that for the purpose of comparison with the current paper, given (S, R) then if we consider instantiations of the form (S, R, I), where I is an instantiation into labelled formulas of some labelled deductive system, i.e. $I : S \mapsto$ LDS, then such a system would generalise and include both approaches; the Arieli–Strasser approach and the Besnard–Hunter approach. This, however, is the subject for a new paper.

We concldue by looking at Figure 74, which explains the situation of Example F.1.

The preferred extensions for this figure are the following

1. $\neg b = \neg a = 0, a = 1, b = 0, \neg c = 1, c = 0$
2. $\neg b = \neg a = 0, a = 1, b = 0, \neg c = 0, c = 1$
3. Same as (1), with $a = 0, b = 1$

4. Same as (2), with $a = 0, b = 1$.

Example F.2 *Let us do Example 17 from Arieli and Strasser [29]. This is to show how simple the labelled approach is compared with the Gentzen formulation. Gentzen systems were invented to prove the consistency of arithmetic. It is risky to take off the shelf tool designed for one purpose and apply it to another purpose; such a move requires proper justification.*

The data of this example is:

(t1) m

(t2) a

(t3) $m \to \bigcirc\neg f$

(t4) $m \wedge a \to \bigcirc f$.

The meaning of the normative $\bigcirc f$ *and* $\bigcirc\neg f$ *is not important here. It is sufficient to note that they attack each other. We can derive:*

1. $(t_3, t_1) : \bigcirc\neg f$
2. $(t_1, t_2) : m \wedge a$
3. $(t_3, (t_1, t_2)) : \bigcirc f$

(1) and (3) attack each other. They have labels telling us how they were proved and one can define an attack relation sensitive of the labels. In LDS we call this "flattening". See [34].[16]

In fact, the attack relation can be a relation R on labels. So S can be the labels and $R \subseteq S \times S$. *This is OK since the labels contain the information of the proofs, including the proved formulas. In fact, in my book on LDS [34], I use many times the formulas themselves as their own labels. So* $\bigcirc\neg f$ *is labelled by* (1^*) *and* $\bigcirc f$ *by* (3^*)*, where*

(1^*) $(m \to \bigcirc\neg f, m) : \bigcirc\neg f$

(3^*) $(m \wedge a \to \bigcirc f, (m, a)) : \bigcirc f$

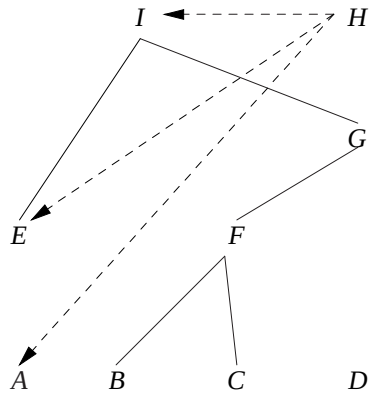
Now compare this with the Gentzen formulation in Figure 3 of [29]. We reproduce it as Figure 75 (the horseshoe is classical implication and the double arrow is the main symbol for the Gentzen sequent).

The problem is not so much the complexity of the representation. The problem is that proofs in a Gentzen system do not flow with the implication, while the human argument does follow the implication. Arieli and Strasser are just using Gentzen as a meta-level deductive machine!

Figure 76 gives the argumentation network the way we build it. We simplified $m \wedge a \to \bigcirc f$ *as* $(m, a) \to \bigcirc f$*, so we avoid conjunction. There is nothing special to this move.*

You can see that paths indicate chains of implications, very intuitive the way we think of it!

[16] We may have in LDS that we can prove $t_i : X$ and also $s_j : \neg X$, yielding multiple bilateral attacks between X and $\neg X$ from different proofs. The Flattening process decides, based on $\{t_i, s_j\}$ whether X or $\neg X$ has the upper hand.



$$
\begin{array}{lcl}
\hat{A} & = & m \supset \bigcirc\neg f \Rightarrow m \supset \bigcirc\neg f \\
\hat{B} & = & m \Rightarrow m \\
\hat{C} & = & a \Rightarrow a \\
\hat{D} & = & (m \wedge a) \supset \bigcirc f \Rightarrow (m \wedge a) \supset \bigcirc f \\
\hat{E} & = & m, m \supset \bigcirc\neg f \Rightarrow \bigcirc\neg f \\
\hat{F} & = & m, a \Rightarrow m \wedge a \\
\hat{G} & = & m, a, (m \wedge a) \supset \bigcirc f \Rightarrow \bigcirc f \\
\hat{H} & = & m, a, (m \wedge a) \supset \bigcirc f \Rightarrow \neg(m \supset \bigcirc\neg f) \\
\hat{I} & = & m, a, m \supset \bigcirc\neg f, (m \wedge a) \supset \bigcirc f \Rightarrow \bigcirc\bot.
\end{array}
$$

Figure 75:

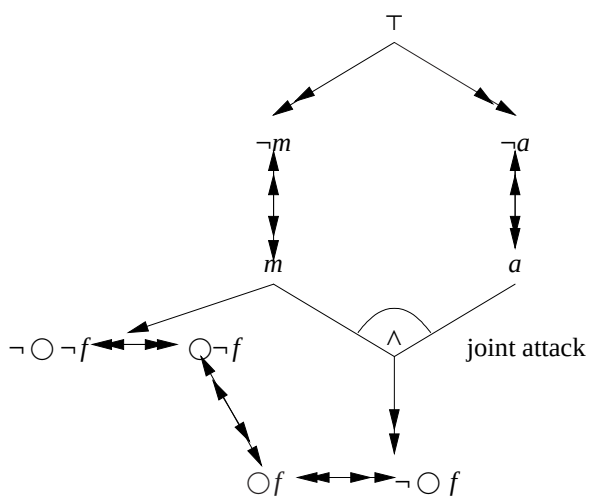


Figure 76:

We conclude this Appendix by quoting a response from C. Strasser and O. Arieli.

Begin quote.

Thanks for referring to our work in the above-mentioned paper. Below are some comments and a response to several issues.

1. First, since Besnard and Hunter's (BH) formalism is mentioned in the same appendix, let us emphasize the differences between our approach and theirs. According to BH, an argument is a pair $\langle \Gamma, \psi \rangle$, where Γ is a subset-minimal consistent set of propositional formulas that entails according to classical logic the propositional formula ψ. In our approach none of these is assumed: languages other than the propositional one may be used, Γ need not be consistent nor minimal, and the underlying logic need not be classical logic. Another difference is that we enhance the calculus of the base logic by elimination rules (see Item 3 below).
2. The general view on arguments, as indicated in the previous item, may serve as a justification for our choice to incorporate sequents in our framework: once an underlying logic $\mathcal{L}$ is fixed, an argument $\langle \Gamma, \psi \rangle$ in our sense is an indication that logically follows, according to $\mathcal{L}$, from Γ. If a sound and complete sequent calculus C exists for $\mathcal{L}$ it serves as a syntactical tool for constructing complex arguments from simpler ones. Such a mechanism must accompany, either implicitly or explicitly, any structural (logic-based) argumentation system, so we are not sure that we follow the criticism in this case. Moreover, to some extent (we still have to check this more carefully), and in the notations of your paper, a labeled formula $t_\Delta : \psi$ may be associated with the sequent $\Delta \Rightarrow \psi$ (the way we encode arguments in COMMA'14), or with a concrete proof of a sequent $\Delta \Rightarrow \psi$ (the way we encode arguments in DEON'14 — see also Item 6 below).
3. We do not agree with the claim that our use of sequent-based Gentzen-style systems is purely a 'meta-level deductive machine', as in addition to deductions, aimed at systematically constructing arguments, we also have sequent-based rules for *eliminating* arguments. Thus, sequents are not only meta-leveled deducible objects, but they are syntactical entities that may be retracted as well. This has two implications. First, the (enhanced) calculus does not only produce the arguments for the argumentation framework but also the attacks. Second, this tighter link between the calculus of the base logic and argumentation frameworks (as compared to the BH-approach) allows to define a machinery for automated deduction on the basis of dynamic proofs (see our COMMA'14 paper and the literature on adaptive logics). It follows that derivations are more complicated structures than 'ordinary' proofs in Gentzen-type systems, which also allow for non-monotonic reasoning.

4. Example F.1: It is noted that ‘the obvious revision is to delete either (t1) or (t2)’. One may argue that eliminating both (t3) and (t4), thus inviting both aunties as well as Caterina, is also a plausible revision. This view, which is more in-line with a paraconsistent view of 1 the state of affairs, in which all the assertions in a theory are treated uniformly, may be supported by our setting, depending on the predefined logic and the attack rules.
5. We are not sure that we understand what do you mean by ‘flowing with implication’. Whatever this property may be, undesirable properties may be lifted by modifying the corresponding proof system (and maybe also by changing the underlying logic). As we indicated before, this is fully supported by our approach.
6. Please note that in the DEON’14 paper an argument is the whole proof of ψ from Γ (and so attacks may be on subproofs). This is similar to the way the ASPIC system views arguments. In the COMMA’14 paper we adopted a simpler view, in which an argument is simply a sequent (or, alternatively, only the ‘top sequent’ of a proof). Both representations of arguments seem to be different than the way that argument are encoded in your Dunglike digraphs, where vertices are propositions. In view of this it is also difficult to directly compare the two approaches in terms of representational complexity/transparency as you seem to do at the end of appendix.

End quote.