Computational Models of Argument
S. Modgil et al. (Eds.)

doi:10.3233/978-1-61499-906-5-109

Relating the ANGELIC Methodology and ASPIC+

Katie ATKINSON [a,1], Trevor BENCH-CAPON [a]
[a] *Department of Computer Science, University of Liverpool, UK*

Abstract. We relate the ANGELIC methodology for acquiring and encapsulating domain knowledge to the ASPIC+ framework for structured argumentation. In so doing we hope to facilitate the building of applications in concrete domains by linking a successful methodology to a proven theoretical framework. We use an example from the ASPIC+ literature to illustrate the relationship.

Keywords. ASPIC+, ANGELIC methodology, Structured Argument

1. Introduction

The ANGELIC methodology [2] was developed in order to acquire and encapsulate knowledge related to legal domains. It has been evaluated both relative to well-known academic domains [2] and, in collaboration with a large law firm, in relation to the practical domain of Noise Induced Hearing Loss [4]. In common with most software engineering methodologies (e.g. [15] and [12]) the methodology is based on its target documentation. The role of the documentation is to structure and modularise the domain, to guide the software engineer as to what information is required and whether it is complete, to record the domain structure and provide information for use by experts for verification, programmers realising the design, and future software engineers who want to reuse, enhance or refine the system. ANGELIC uses Abstract Dialectical Frameworks (ADFs) [10] as the chosen documentation structure. As explained in [2] ADFs provide an excellent way of modularising the system and encourage and support a functional decomposition design approach. In ANGELIC the ADF essentially performs the role of ER diagrams in database methodologies such as [12]. Moreover by associating additional properties such as provenance, node description and domain with the ADF nodes [1], the resulting document (Table 1 provides an example), can also serve as a Data Dictionary.

A methodology is essential for the building of actual, reasonably sized, applications. Early expert systems fell foul of the *Knowledge Engineering Bottleneck* [14], and it remains a problem today, which partly accounts for the increasing popularity of machine learning approaches. These, however, have problems in domains such as law in which the explanation is more important than the answer, the interpretation of the data is constantly subject to reinterpretation and where the data may be contaminated by bias and prejudice (often unconscious). For some applications the bitter pill of knowledge acquisition must be swallowed, and a methodology to acquire and record the knowledge is needed.

[1] Corresponding Author: K.M.Atkinson@liverpool.ac.uk

A methodology, however, does require a formal underpinning, in the way that relational algebra underpins database systems [13]. Because the target output of legal applications is very close to the structured arguments and attacks that make up the ASPIC+ framework [16], we would like to use ASPIC+ as this formal underpinning. We will therefore relate the structures produced by the ANGELIC methodology to ASPIC+. The existence of a methodology which can be used to build substantial applications should also facilitate the construction of argument based systems in usefully sized realistic applications. Our contribution is thus twofold: ANGELIC is given a sound basis, and a means of developing concrete applications with well understood properties is provided.

Because this is a short paper, we will assume familiarity with ANGELIC [2], ADFs [10] and ASPIC+ [16]. We will not reproduce formal definitions, and readers unfamiliar with them are referred to the original papers. Some definitions are also given in a technical report [6], which is a longer version of this paper.

2. Use of Abstract Dialectical Frameworks

ADFs are defined in [10]. They comprise a set of *nodes* representing statements, a set of *links* between the nodes and a set of *acceptance conditions*, where the acceptance conditions of nodes are expressed solely in terms of their children, providing the required modularisation. Although in [9] ADFs have been extended to accommodate statements with truth values in range $[0,1]$, and ANGELIC now also allows this [8], here we will only consider Boolean valued statements (as used in ASPIC+ in [16]). In ANGELIC the statements relate to issues, intermediate factors and base level factors as found in CATO's factor hierarchies [5], and subsequent developments using factor based reasoning [7][2]. In ANGELIC the acceptance conditions take the form of a set of sufficient conditions for ascription of 0 (false) or 1 (true) to the node together with a default should none of the sufficient conditions be satisfied. The explicit attribution of false obviates the need for negation as failure, and the default can be either 1 or 0 to reflect the burden of proof. The conditions are prioritised by their order and the existence of the default means that, taken together, the sufficient conditions and default provide a necessary condition, providing Clark completion [11]. Note also that the ADFs produced by ANGELIC are cycle free and exhibit a tree structure, thus allowing interpretation with grounded semantics.

3. Relation to ASPIC+

The structure of arguments is described formally and in detail in ASPIC+ [16]. Definitions of *argumentation system* (a triple of a language, a set of rules (partitioned into strict and defeasible), and a function to name defeasible rules), and *argument theory* (a pair comprising an argument system and a knowledge base (partitioned into *axioms* and contingent *facts*) are given in [16]. Arguments are defined relative to an argumentation theory and serve to chain applications of the inference rules into inference trees. Arguments have an argument/subargument structure and properties including: conclusion, premises,

[2]Essentially the *issues* provide the questions that must be answered to decide the case, and the base level factors represent the *legal facts* of a particular case. The *intermediate factors* are legal concepts relating the facts to the issues: e.g. [3].

Example 3.7 Consider a knowledge base in an argumentation system with $\mathcal{L}$ consisting of $p, q, r, s, t, u, v, w, x, d_1, d_2, d_3, d_4, d_5, d_6$ and their negations, with $\mathcal{R}_s = \{s_1, s_2\}$ and $\mathcal{R}_d = \{d_1, d_2, d_3, d_4, d_5, d_6\}$, where

d_1: $p \Rightarrow q$ $\quad$ d_4: $u \Rightarrow v$ $\quad$ s_1: $p, q \rightarrow r$
d_2: $s \Rightarrow t$ $\quad$ d_5: $v, x \Rightarrow \neg t$ $\quad$ s_2: $v \rightarrow \neg s$
d_3: $t \Rightarrow \neg d_1$ $\quad$ d_6: $s \Rightarrow \neg p$ $\quad$ $A = \{p\}$. $F = \{s,u,x\}$

Figure 1. Example 3.7 of [16]

subarguments, the last (top) rule and the defeasible rules used. The main example of [16] is shown in Figure 1. Table 1 represents the example as an ADF. Each proposition is a node of the ADF, and the rules are incorporated in the acceptance conditions for the consequent. A diagrammatic version of the ADF is given in [6].

An ASPIC+ Argumentation Theory requires a language, Here the language is based on the nodes of the ADF. Statements take the form *Node(TruthValue)*. Since here only Booleans are used, we can, where convenient, omit the numeric truth value, so that $p(1)$ becomes p and $p(0)$ becomes $\neg p$

We should point to some differences between the ASPIC+ representation and an ADF representation using the ANGELIC methodology.

- An ADF produced using ANGELIC includes explicit default rules in the acceptance conditions, effectively completing the database [11]. Explicit defaults have the advantage of allowing defaults of true as well as false and supply a rule which can be referred to if desired.
- ANGELIC normally allows only base level factors (those for which there are no strict or defeasible rules) to appear in the sets A and F. But in 3.7, p and s occur in both rules and premises. This is because ANGELIC was designed specifically for modelling bodies of legal case law, which distinguish between legal facts and the intermediate concepts derived from them [3]. In a legal case the facts are disputable in a court of first instance and so will appear in F, but in higher courts, where the facts cannot be challenged, they will move to A. If this restriction is applied, non base level factors will not need to be tested for their presence in A and F, and conditions such as SR2 and PR1 can be omitted.

3.1. Arguments and Attacks

The definition of arguments in ASPIC+ [16] allows an argument for a node if it is an axiom or a fact, or if it is the conclusion of a rule the premises of which can also be justified by an argument. In ANGELIC, each test in the acceptance conditions is potentially the last step of an argument for the corresponding node, with the truth or falsity of the node as conclusion and the appropriate children as premises. The test used will be ASPIC+'s *top-rule*, while the sub arguments will be those giving the status of the children used in the test. An argument for r is shown as argument A at the top left of Figure 2. It is grounded in the axiom p (argument A1). From this we can derive q using PR2 (argument A2). Note that we establish that t is false by default: this is not shown in Figure 2. Now we can derive r using SR1 (argument A3).

Table 1. ADF for example 3.7 in [16]. "v" is a variable, either 0 or 1. *A* is the set of axioms, *F* is the set of facts. All the nodes are taken from the same domain, and the provenance of the acceptance condition is shown by giving the particular rule named in Example 3.7. SRn is a strict rule, PRn is a presumptive (defeasible) rule and DRn is a default rule.

ID	Name	Domain	Children	Acceptance conditions	Provenance
N1	r	3.7	(p,q)	SR1: r(1) if p(1) & q(1) SR2: r(v) if r(v) in A PR1: r(v) if r(v) in F DR1: r(0)	s1
N2	q	3.7	p,t	SR3: q(v) if q(v) in A PR2: q(1) if t(0) and p(1) PR3: q(v) if q(v) in F DR2: q(0)	d1, d3
N3	t	3.7	s , (v,x)-	SR4: t(v) if t(v) in A PR4: t(0) if v(1) & x(1) PR5: t(1) if s(1) PR6: t(v) if t(v) in F DR3: t(0)	d5 d2
N4	v	3.7	u	SR5: v(v) if v(v) in A PR7: v(1) if u(1) PR8: v(v) if v(v) in F DR4: v(0)	d4
N5	p	3.7	Axioms, s-	SR6: p(v) if p(v) in A PR9: p(0) if s(1) PR10: p(v) if p(v) in F DR5: p(0)	d6
N6	s	3.7	Facts, v-	SR7: s(v) if s(v) in A SR8: s(0) if v(1) PR11: s(v) if s(v) in F DR6: s(0)	s2
N7	u	3.7	Facts	SR9: u(v) if u(v) in A PR12: u(v) if u(v) in F DR7: u(0)	
N8	X	3.7	Facts	SR10: x(v) if x(v) in F PR13: x(v) if x(v) in F DR8: x(0)	

There are three kinds of attack in ASPIC+ as defined in [16]. An attacker *A* of *B* can *undercut* (defeat a rule used in *B*), *rebut* (conclude the negation of *B*'s conclusion) or *undermine* (show a premise used in *B* to be false). Figure 2 shows *B* undercutting *A*, *D* rebutting *B*2 and *C* undermining *B*. For rebuttals, arguments based on strict rules cannot be rebutted and defeasible and default rules can be rebutted by arguments which conclude the negation of their conclusion. Strict rules successfully rebut both defeasible rules and defaults, and defeasible rules normally successfully rebut defaults. So for a debatable rebuttal to arise, there needs to be two defeasible rules in the acceptance condition of a

node, both of which have their bodies satisfied and which ascribe different values to the node statement. The rebuttal in the example is on t: v, s and x can all be justified and so both PR4 (justifying the acceptance of t) and PR5 (justifying the rejection of t) can be deployed in arguments. This necessitates a choice between them. Both ASPIC+ and ANGELIC resolve this using rule priorities. In ANGELIC the priority is given by the order of the conditions.

Underminers are found by tracing back through the arguments until we find a challenge to a premise. The underminer in the example is on s. Although s has been assumed to be a fact, we have a strict argument for $\neg s$ based on SR7, which will succeed (unless C is defeated). Had the arguments both been based on defeasible rules, we would need to resort to priorities. We might well expect a general picture of facts taking precedence over rules, or vice versa, depending on the particular domain.

Both of these kinds of attack are relatively straightforward. Undercutters are, however, more interesting. In Table 1 we have represented both of the rules d_1 and d_4 in a single rule, PR2. From a logical point of view this is fine: given p, we can deduce q (t being false by default), so d_1 can be applied. If, however, t turns out to be true, the antecedent of PR2 is no longer true, and so d_1 can no longer be applied. In the example d_4 can be used to prevent d_1 being applied. But we have an asymmetry between p and t: while the truth of *t undercuts* an argument based on PR2, the falsity of *p undermines* it. The reason is that undercutting is an epistemological or dialectical notion rather than a logical one. If an antecedent is established using a default rule, should the default be overridden the rule is no longer applicable: i.e. it is undercut. Whereas if the antecedent is not the default value (p defaults to false according to DR5) it must be shown before the rule can be applied, and so needs an underminer to prevent the application of the rule.

This relates to the legal distinction between *probanda* and *non refutanda* at the heart of [17]. The idea is that to establish a claim, the claimant must prove certain propositions, but other propositions may be presupposed unless the opponent can show the presuppositions to be false. This is exactly what defaults enable. Essentially the distinction is one of who has the burden of proof. The claimant has the burden of proof for the *probanda*, but may use defaults for the *non refutanda*, putting the burden of proof for these on the opponent who must come up with positive reasons to disbelieve the *non refutanda*. It is often said that law is expressed as general rules and exceptions: we can see the *probanda* as the antecedent of the rule, and the *non refutanda* as the exceptions. We could express this by extending our notation and rewrite PR2 as: *PR2a: q(1) if p(1) : t(0)* with the colon read as *presupposing*.

Perhaps the most interesting aspect of this account is that it gives a clear explanation of why in ASPIC+ conflicting arguments arising from rebuttal and undermining are resolved using preferences or priorities, whereas undercutters always succeed in defeating the argument they attack. We can now see that while rebuttals and underminers involve choosing between two arguments, with undercutters there is no choice: the undercutting argument activates the exception, destroying the undercut argument. That the exception is preferred is already decided: if the exception were not preferred it would not appear in the representation at all, since it would have no effect. This links with the point made in relation to value-based reasoning in [18] and [8], that values not only allow for a preference between rules in terms of the values they promote, but also justify the inclusion of some of the antecedents in rules, in terms of the values they allow to be considered. While the first use relates to rule priorities, the second relates to exceptions.

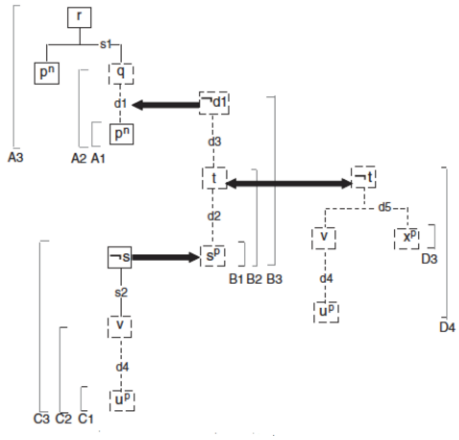

Figure 2. Argument for R with subarguments and attacks from [16]

One advantage of the modularisation provided by ANGELIC is that potential undercutters and rebuttals can be identified by considering a single node. As discussed above, undercutters arise from a rule with an exception: thus the undercut of argument A2 in Figure 2 arises from the exception rule PR2a. If there are no rules with exceptions of this sort in the acceptance conditions for the conclusion, the argument for that conclusion cannot be undercut. Rebuttals arise when the acceptance conditions contain non strict rules for conflicting degrees of acceptance: arguments (arising from defeasible rules) and presuppositions (arising from default rules) may be rebutted. The rebutting arguments in Figure 2 are B2 and D4, which conclude t and $\neg t$ respectively. If we examine the acceptance conditions for t in Table 1, we see we have two defeasible rules (PR4 and PR5) with contrary conclusions. The possibility of rebuttal only arises if the acceptance conditions for a node contain two defeasible/default rules for a statement with conflicting truth values. If one of the rules is a default rule, the presupposition may be considered rebutted. If neither is a default, we resolve the conflict using priorities, either explicit as in ASPIC+ or implicit through the order of the conditions as in ANGELIC.

Underminers such as the undermining of B by C3 in Figure 2 arise when an antecedent of a rule used to establish a statement can be defeated (by a strict rule) or successfully rebutted (by a preferred defeasible rule). Thus t was established using PR5, which requires s to be true. Looking at the acceptance conditions for s, we see we have a (strict) rule to establish that s is false. This, therefore is a potential underminer, which can be actualised by establishing v using PR7 and the fact that u. To find underminers we have to work back through the child nodes.

ANGELIC does not enforce closure under contraposition, as recommended in [16], but it does allow it, so additional conditions can be added where required and appropriate. Thus we could close SR8 under contraposition by adding *v(0) if s(1)* to N4 in Table 1, or SR1 by adding *p(0) if r(0) and q(1)* to N5 and *q(0) if r(0) and p(1)* to N2. It might be desirable to include this as a routine step in ANGELIC.

4. Animating the ADF

Given the ADF in Table 1, it is a straightforward matter to produce a Prolog implementation (for a full description, with code, see [6]). The acceptance conditions form a set of clauses (a Prolog *procedure*) for determining the truth value of the statement represented by the node. We can thus move from the acceptance conditions to Prolog code by a simple rewriting to put them in Prolog syntax. Alternatively we could write the conditions in Prolog syntax from the outset. In Prolog, the order of the clauses determines the priority of the clauses, and this is also true of ANGELIC, but not of ASPIC+. Note that this means that in Table 1, we have a fixed preference order of:

- default rules $\prec$ facts $\prec$ defeasibe rules $\prec$ strict rules $\prec$ axioms

This seems reasonable for most cases, but there could be applications where a different order was desirable: for example to prefer the asserted facts to the conclusions of defeasible rules. We can also add some explanatory tags to provide explanations: e.g for strict rules: *statement is true/false because statements in clause body hold, by rule name*. This will provide an explanation of arguments such as A1,A2,A3 in Figure 2. But we also wish to alert the user to potential attacks. Thus if there is a potential undercutter as in PR2 in Table 1 we extend the tag to get *q is true because p is true by PR2, but can be undercut if t is true* We can also identify potential rebuttals, where there are multiple defeasible rules, as in the conditions for *t* in Table 1, and add tags for these also [6].

In ANGELIC preferences are between *rules*. In ASPIC+, however, preferences are between *arguments*, and so the relevant preference in Figure 2 is between arguments B2 and D4. ASPIC+ proposes two principles for assessing the strength of arguments in [16]: *last*-link and *weakest*-link principles. In *last*-link arguments are compared according to the preference for the defeasible rules used at the highest level of the proof tree. Weakest-link considers all the defeasible elements used in the argument, and compares arguments on the weakest of these. ANGELIC is closer to last-link, but all comparisons are within the node itself, and so the last rules are compared, whether they are strict or defeasible, since the preferences are determined by the ordering in the acceptance conditions. Although ANGELIC has the principle implicit, and uses it to determine the first answer returned by the program, requesting further answers will reveal all the defeasible elements, and so users may apply the ASPIC+ principles if they so desire. Thus in the argument for *r* (see the output in [6]), although the last rule is the strict rule SR2, the last defeasible rule (PR2) and all defeasible elements (PR2, PR4, PR7 and the assertions of *u* and *x*) are also shown, so that the weakest link can be identified. In [16] it is argued that sometimes one principle is better and at other times the other is more suitable. ANGELIC was developed specifically for use with law, and there it may be that the last-link principle is preferred (this does seem to be the way preferences work in, for example [18]). Indeed, in [16] itself there is a suggestion that last-link may be more suitable for legal (or more generally, normative) applications, while weakest-link is more useful in empirical applications (p44). This question merits further consideration.

5. Concluding Remarks

The main purpose of this paper was to show the relation between the output from the ANGELIC methodology [2] and ASPIC+ [16]. The aim of ANGELIC is to assist with

the development of concrete legal applications by the provision of a means of structured acquisition and encapsulation of knowledge, and provision of a modular design to assist with program development and maintenance. For although the argumentation in ASPIC+ is well structured, the same cannot be said for the Argumentation Theory used in the argumentation. The result is that ANGELIC, which has proved to be a successful methodology for supporting domain analysis ([2], [4]) can now benefit from the theoretical underpinning of the ASPIC+ framework, and the formal understanding provided by ASPIC+ can be readily used in practical applications. In doing so we have identified a number of issues for future work:

- Whether some principles can be laid down for the use of last-link as opposed to weakest-link as a way of choosing between arguments. Of particular interest to us is whether the suggestion in [16] that last-link is the more appropriate for legal domains can be substantiated.
- Whether a clean integration with meta-level argumentation to allow reasoning about preferences is possible for ASPIC+, ANGELIC, or both.

References

[1] L. Al-Abdulkarim, K. Atkinson, S. Atkinson, and T. Bench-Capon. Angelic environment. In *Proceedings of the 16th International Conference on AI and Law*, pages 267–269. ACM, 2017.

[2] L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. A methodology for designing systems to reason with legal cases using Abstract Dialectical Frameworks. *Artificial Intelligence and Law*, 24(1):1–49, 2016.

[3] L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. Statement types in legal argument. In *Proceedings of JURIX 2016: The Twenty-Ninth Annual Conference*, pages 3–12, 2016.

[4] L. Al-Abdulkarim, K. Atkinson, T. Bench-Capon, S. Whittle, R. Williams, and C. Wolfenden. Noise Induced Hearing Loss: An application of the Angelic methodology. In *Proceedings of JURIX 2017*, pages 79–88, 2017.

[5] V. Aleven. *Teaching case-based argumentation through a model and examples*. PhD thesis, University of Pittburgh, 1997.

[6] K. Atkinson and T. Bench-Capon. Relating the ANGELIC Methodology to ASPIC+. Technical report, University of Liverpool Computer Science, ULCS-18-001, 2018.

[7] T. Bench-Capon. HYPO's Legacy: Introduction to the virtual special issue. *Artificial Intelligence and Law*, 25(2):205–250, 2017.

[8] T. Bench-Capon and K. Atkinson. Dimensions and values for legal CBR. *Proceedings of JURIX 2017*, pages 27–32, 2017.

[9] G. Brewka. Weighted Abstract Dialectical Frameworks. In *Workshop on Argument Strength*, page 9. Ruhr University, Bochum, 2016.

[10] G. Brewka, S. Ellmauthaler, H. Strass, J. P. Wallner, and S. Woltran. Abstract Dialectical Frameworks revisited. In *Proceedings of the Twenty-Third IJCAI*, pages 803–809. AAAI Press, 2013.

[11] K. L. Clark. Negation as failure. In *Logic and data bases*, pages 293–322. Springer, 1978.

[12] T. M. Connolly and C. E. Begg. *Database Solutions: A step-by-step guide to building databases*. Pearson Education, 2004.

[13] C. J. Date. *An introduction to database systems*. Pearson Education India, 2006.

[14] E. A. Feigenbaum. Knowledge engineering. *Annals of the NY Academy of Sciences*, 426:91–107, 1984.

[15] M. A. Jackson. *Principles of program design*. Academic Press London, 1975.

[16] S. Modgil and H. Prakken. The ASPIC+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62, 2014.

[17] G. Sartor. Defeasibility in legal reasoning. In Z. Bankowski, I. White, and U. Hahn, editors, *Informatics and the foundations of legal reasoning*, pages 119–157. Springer, 1995.

[18] T. Zurek and M. Araszkiewicz. Modeling teleological interpretation. In *Proceedings of the Fourteenth International Conference on AI and Law*, pages 160–168. ACM, 2013.