Relating the ANGELIC Methodology to ASPIC+

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

Abstract. In this paper 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, especially law, by linking a successful methodology to a proven theoretical framework. We use an example from the ASPIC+ literature to illustrate the relationship, and as further illustration provide an executable program able to support argument preparation. Finally we identify several questions for future work.

Keywords. ASPIC+, ANGELIC methodology, Structured Argument

1. Introduction

The ANGELIC methodology [1] was developed in order to acquire and encapsulate knowledge related to legal domains. It has been evaluated both relative to well-known academic domains [1] and, in collaboration with a large law firm, in relation to the practical domain of Noise Induced Hearing Loss [3]. ANGELIC produces Abstract Dialectical Frameworks (ADFs) [9] to structure the domain knowledge: the represents the design documentation which can form the basis of implementations to support a range of tasks [3] in the chosen domain. The nodes of an ADF, however, represent statements rather than arguments, and in legal domains especially it is the arguments we are interested in. Knowing whether a statement is true or false is typically of no use without a justification. In this paper we will explore the relation between the ADFs produced by ANGELIC (which are *trees*, and which express acceptance using a set of individually sufficient and jointly necessary conditions) and structured argumentation, in particular Toulmin's scheme [16] and ASPIC+ (for which we will use [13]). The purpose of the work is twofold: to facilitate the use of theoretical research on argumentation in substantial real world applications, by linking it to a methodology which is capable of tackling such applications, and to provide a sound theoretical basis to underpin the methodology.

2. Abstract Dialectical Frameworks

ADFs are defined in [9]. They comprise a set of *nodes* representing statements, a set of *links* between the nodes and a set of *acceptance conditions*, where the acceptance con-

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

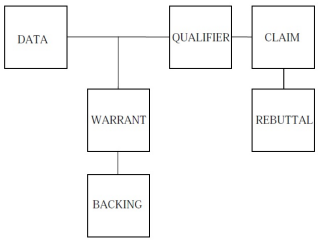

Figure 1. Toulmin's Argument Scheme

ditions of nodes are expressed in terms of their children. In [9] the statements are three valued (true, false and undecided), but this has subsequently been generalised [8] to arbitrarily truth-valued statements, In ANGELIC the statements relate to issues, intermediate factors and base level factors as found in CATO's factor hierarchies [4], and subsequent developments using factor based reasoning [5][2]. Although standard Boolean propositional logic sufficed for the factor based reasoning used in the applications modelled in [1], subsequent work (e.g. [14] and [2]) has found that some notion of magnitude or degree is often required to capture the nuances of legal reasoning. Currently ANGELIC can use either Booleans [1] or truth values which map to real numbers in the range 0 to 1 and interprets logical connectives in the manner of fuzzy logic [17].

The links show which nodes are used to determine the acceptability of other nodes, so that the acceptability of a parent node is determined solely by its children. It is this feature which provides the strong modularisation which is important from a software engineering point of view, and which is lacking in many rule-based systems. The acceptance conditions for a node state how precisely its children relate to that node. As described in [1], ANGELIC originally represented the acceptance conditions for non-leaf nodes as a set of individually sufficient and jointly necessary conditions for the parent to be accepted or rejected, but this has been subsequently extended to accommodate *degrees of acceptance*, as described below. Additionally, the truth conditions are also associated with their provenance, such as statutes, commentaries and particular cases.

3. Relation to Toulmiin's Scheme

Toulmin's argument scheme, shown in Figure 1, has been a popular way of showing the structure of arguments since its introduction in [16]. We will briefly relate this scheme to our ADFs. Readers who want an example ADF can refer to Table 1 below.

Each individual acceptance condition instantiates the scheme. The data is the set of children used in the acceptance condition, the conclusion is the node itself, and the qualifier is the degree of acceptance attributed by the acceptance condition, with 1 unqualifiedly true and 0 unqualifiedly false. The acceptance condition as a whole is the warrant and its provenance is the backing. Finally the rebuttal is the disjunction of other acceptance conditions attributing a different degree of acceptance. Since Toulmin does

[2]Essentially the issues provide the questions that must be answered to come to a decision in 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.

not chain his schemes, arguments are not inter-related, and so only a single node need be examined.

4. Relation to ASPIC+

The structure of arguments is described more formally and in greater detail in ASPIC+ [13]. An *argumentation system* is defined in [13] as a triple $AS =< \mathscr{L}, \mathscr{R}, n >$ where

- $\mathscr{L}$ is a logical language closed under negation;
- $\mathscr{R} = \mathscr{R}_s \cup \mathscr{R}_d$ is a set of strict and defeasible inference rules of the form ϕ_1 ... $\phi_n \rightarrow \phi$ and ϕ_1 ... $\phi_n \Rightarrow \phi$ respectively, where ϕ_i, ϕ are meta-variables ranging over wffs in $\mathscr{L}$ and $\mathscr{R}_s \cup \mathscr{R}_d = \emptyset$;
- n is a partial function such that $n : \mathscr{R}_d \longrightarrow \mathscr{L}$.

As stated in [13], an *argument theory AT* is a pair of an *argument system AS* and a *knowledge base K* in *AS*. Now arguments are defined relative to an argumentation theory $AT =< AS, K >$, and serve to chain applications of the inference rules from *AS* into inference trees, starting with elements from the knowledge base *K*. In what follows, for a given argument, the function `Prem` returns all the formulas of *K* (called premises) used to build the argument, `Conc` returns its conclusion, `Sub` returns all its sub-arguments, `DefRules` returns all the defeasible rules of the argument and `TopRule` returns the last inference rule used in the argument. Now an argument *A* is

1. ϕ if $\phi \in K$ with: `Prem`(A) = $\{\phi\}$, `Conc`(A) = ϕ, `Sub`(A) = $\{\phi\}$, `DefRules`(A) = $\emptyset$, `TopRule`(A) = undefined.
2. A_1, ... $A_n \rightarrow \psi$ if A_1, ... A_n are arguments such that there exists a strict rule for which `Conc`(A_1), ..., *Conc*(A_n) $\rightarrow \psi$ in $\mathscr{R}_s$.
   `Prem`(*A*) = `Prem`(A_1) $\cup$ ... $\cup$ `Prem`(A_n),
   `Conc`(*A*) = ψ,
   `Sub`((*A*) = `Sub`((A_1)) $\cup$ ... $\cup$ `Prem`(A_n) $\cup \{A\}$.
   `DefRules`(*A*) = `DefRules`(A_1) $\cup$ ... $\cup$ `DefRules`(A_n).
   `TopRule`(*A*) = `Conc`(A_1), ..., `Conc`(A_n) $\rightarrow \psi$.
3. A_1, ... $A_n \Rightarrow \psi$ if A_1, ... A_n are arguments such that there exists a defeasible rule for which `Conc`(A_1), ..., *Conc*(A_n) $\rightarrow \psi$ in $\mathscr{R}_d$
   `Prem`, `Conc` and `Sub` are as (2).
   `DefRules`(*A*) = `DefRules`(A_1) $\cup$ ... $\cup$ `DefRules`(A_n) $\cup$ {`Conc`(A_1), ..., `Conc`(A_n) $\Rightarrow \psi$}
   `TopRule`(*A*) = `Conc`(A_1), ..., `Conc`(A_n) $\Rightarrow \psi$.

The main example of [13] is shown in Figure 2. Table 1 represents the example as an ADF. Figure 3 shows an argument from this example. Figure 4 is a diagrammatic version of the ADF.

4.1. Argumentation Theory

An ASPIC+ Argumentation Theory requires a language $\mathscr{L}$, a set of rule names, and a function from names to rules expressed in $\mathscr{L}$. Here the language is based on the nodes of the ADF. Statements take the form:

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

$$\begin{array}{lll} d_1: \ p \Rightarrow q & d_4: \ u \Rightarrow v & s_1: \ p, q \rightarrow r \\ d_2: \ s \Rightarrow t & d_5: \ v, x \Rightarrow \neg t & s_2: \ v \rightarrow \neg s \\ d_3: \ t \Rightarrow \neg d_1 & d_6: \ s \Rightarrow \neg p & \end{array}$$

Figure 2. Example 3.7 of [13]

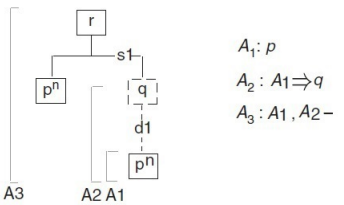

Figure 3. Argument for R in example 3.7 of [13]

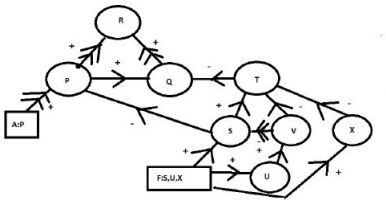

Figure 4. ADF of example 3.7 of [13]. Double arrows indicate a strict relationship.

- *Statement: Node(Degree)*

Degree is a real number in range [0,1]. This can be read as *Node* has a degree of acceptability *Degree*. The acceptance[3] conditions are a set of sufficient conditions for attributing a degree of acceptance to a node, and a default, used when none of the conditions apply. They may take a variety of forms, including AND and OR (both interpreted as in [17]), comparison with a threshold, or some other arithmetic formula, typically used to express trade-offs [6]. These conditions resemble clauses of logic programs (with negation represented by degree 0, obviating the need for negation as failure). This requires two functions:

- *val(statement)* which maps from statement to a real number:*val(Node(d)) = d)*.
- *stat(node, d)* which maps from a node and a real number [0,1] to statement: *stat(node,d) = Node(d)*.

Thus an acceptance condition may take forms such as:

- $Parent(1) \leftarrow val(Child1(v1) - val(Child2(v2) \geq threshold$
- $stat(parent, v) \leftarrow v = val(Child1(v1)) + val(Child2(v2))$

For simple examples, such as example 3.7 of [13] only Booleans are needed and so only the degrees 0 and 1 are used. Where only Booleans are used, we omit degree, so that $p(1)$ becomes p and $p(0)$ becomes $\neg p$

Conditions representing strict rules and those recognising that the statement is an *axiom* form the elements $\mathscr{R}_s$ (named *SR*1 etc.), while those representing defeasible rules and those recognising that the node may be a contingent *fact* are the elements $\mathscr{R}_d$ (named *PR*1 etc. (P for presumptive)). In the modelling of legal domains in ANGELIC, only the

[3]We continue to use the term *acceptance* conditions even though the condition may be for the rejection of the statement. Acceptability is strictly of the meta-level statement *node has degree d*.

Table 1. ADF for example 3.7 in [13]. "v" is a variable, either 0 or 1.

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)	

base-level nodes will appear in axioms and facts, in which case the corresponding rules for higher level nodes can be omitted. The defaults (one per node) are named *DR*1 etc. These are then distributed across the appropriate nodes. We also include the body of the rule in Table 1).

4.2. Differences between ASPIC+ and an ANGELIC ADF

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

- The ADF includes explicit default rules in the acceptance conditions, so effectively completing the database [10], whereas ASPIC+ uses negation as failure.

This has the advantage of allowing defaults other than false (especially useful if the statements are not Boolean) and supplies a rule which can be referred to if desired.

- ASPIC+ allows any statement to appear in K_n and K_p, whereas ANGELIC normally allows only base level factors (those for which there are no strict or defeasible rules) to appear in the corresponding sets A and F. This is because ANGELIC was designed specifically for modelling bodies of legal case law, which needs to distinguish between legal facts and intermediate concepts [2]. In ANGELIC 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 wiil not need the rules of the form *P(v) if P(v)*$\in A$ and *P(v) if P(v)*$\in F$.

4.3. Arguments

The definition of arguments in ASPIC+ given above is tripartite: the first part defines the base case and states that an argument exists for a node if it is an axiom or a fact. This is also the definition invoked when an ANGELIC default is used: we could see defaults as a third partition of K, but this would not emphasise that *every* node has a default. The other two parts of the definition are recursive: stating that an argument exists for a node if there is a series of subarguments, the last of which uses a rule which has the required node as its conclusion. The difference between parts (2) and (3) depends on whether the relevant rules are strict or defeasible.

Each rule in the acceptance conditions is potentially the last step of an argument for the corresponding node. Now for an argument A the function `Conc(A)` will return the node and the degree assigned by the rule; `Prem(A)` will return the premises of the arguments justifying the conclusion, that it the elements in the body of the rule; `Sub(A)` will return in the arguments justifying the elements in the body of the rule and A itself; `Def(A)` will return the defeasible rules in the arguments justifying the elements in the body of the rule and, the final rule in the chain (which appears in the acceptance condition) will be returned by `TopRule(A)`. An argument for r is shown in Figure 3. It is grounded in the axiom p (argumnent 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 3. Now we can derive r using SR1 (argument A3).

4.4. Attacks

There are three kinds of attack in ASPIC+, as defined in [13]. An attacker A can undercut, rebut or undermine an argument B.

- A undercuts B (on B') iff `Conc`$(A) = \neg n(r)$ for some $B' \in$ `Sub`(B) such that the top rule of B'' is defeasible.
- A rebuts B (on B') iff `Conc`$(A) = \neg\phi$ for some $B' \in$ `Sub`(B) of the form B''_1, ..., B''_n $\Rightarrow \phi$.
- A undermines B (on ϕ) iff `Conc`$(A) = \neg\phi$ for an ordinary premise ϕ of B

Attacks on the argument of Figure 3 are shown in Figure 5. Rebuttals are straightforward. Strict rules cannot be rebutted and defeasible and default rules can be rebutted

by rules with a different degree in the head. 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 degrees 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 $\mathtt{t}$) 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 implicit in the order of the conditions.

Underminers are found by tracing back through the arguments until we find a rebuttal. The underminer in the example is on s. Although s has been assumed to be a fact in K_d, we have a strict argument for $\neg s$ based on SR7, which will succeed. Had the arguments both have 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 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, if the default is 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 is 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 [15]. The idea is that to establish the claim, a claimant must prove certain things, but others may be presupposed, unless the opponent can show the presuppositions to be false. This exactly what default values 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 or proof on the opponent who must come up with a positive reasons to disbelieve the *non refutanda*. It is often said that law is expressed as general rules and exceptions [11]: we can see the *probanda* as the antecedent of the rule, and the *non refutanda* as the exceptions. We can express this by extending our notation. We could 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 two arguments, in undercutters there is no conflict between arguments: the undercutting argument does not conflict with the undercut argument, but activates the exception. 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 [6], that values not only al-

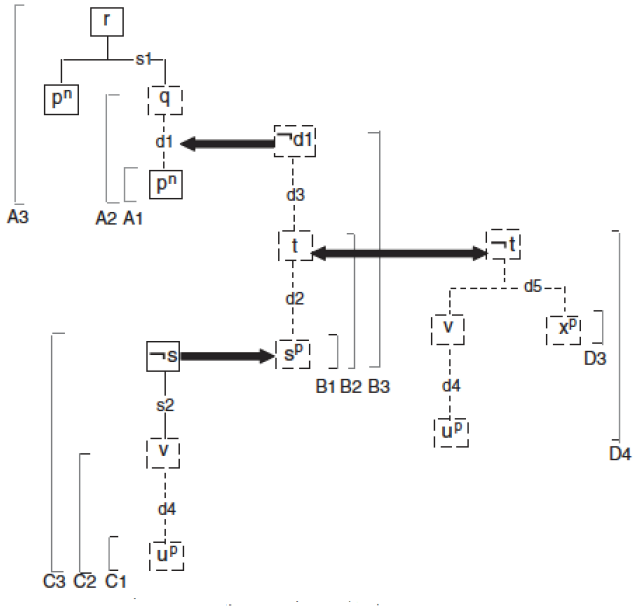

Figure 5: Argument for r with subarguments and attacks

low 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.

4.5. Attacks and the ADF

As discussed in the previous section, undercutters arise from a rule with an exception: thus the undercut of argument A2 in Figure 5 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. In this way both arguments (arising from defeasible rules) and presuppositions (arising from default rules) may be rebutted. The rebutting arguments in Figure 5 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 rules for a statement with conflicting degrees for that statement. 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 5 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.

5. Animating the ADF

Given the ADF in Table 1, it is a straightforward matter to produce a Prolog implementation. 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 $\prec$ assertions $\prec$ 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 assertions to the conclusions of defeasible rules. We now add some explanatory tags to provide explanations:

- For axioms: *statement is true/false because axiom*.
- For strict rules: *statement is true/false because statements in clause body hold, by rule name*.
- For defeasible rules: *statement is presumptively true/false because statements in clause body hold, by rule name*
- For assertions: *statement is true/false because assertion*
- For defaults: *statement is true/false by default*

This will provide an explanation of an argument such as A1,A2,A3 in Figure 5. 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. To draw attention to this we add

- *possible rebuttal from rule name(s)*

to the tag for defeasible rules. Putting all this together, we get the following Prolog code for *t*:

```
t(t):-axioms(L), member(t,L),
      write([t,is,true,because,axiom]),nl.
t(f):-v(t),x(t),
write([t,is,presumptively,false,because,both,v,and,x,are,true,pr4,
      possible, rebuttal,from,pr5]),nl.
t(t):-s(t),write([t,is,presumptively,true,because,s,is,true,pr5,
      possible, rebuttal,from,pr4]),nl.
t(t):-assertions(L), member(t,L),
      write([t,is,true,because,assertion]),nl.
t(f):-write([t,is,false,by,default]),nl.
```

We can now execute the program to get a potential argument for, say, the truth of *r*. The Prolog output has been annotated to show the correspondences with the arguments in Figure 5.

```
1 ?- r(R).
[p,is,true,because,axiom]                              AI
[p,is,true,because,axiom]                              A1
[u,is,true,because,assertion]                          D1
[v,is,true,because,u,is,true,pr7]                      D2
[x,is,true,because,assertion]                          D3
[t,is,presumptively,false,because,both,v,and,x,
     are,true,pr4,possible,rebuttal,from,pr5]          D4
[q,is,true,because,p,is,true,pr2,                      A2
    but,can,be,undercut,if,t,is,true]                  B3
[r,is,true,because,both,p,and,q,are,true,sr1]          A3
R = t
```

Note that *p* appears twice because it is used in both SR1 and PR2. The argument is also extended when compared with Figure 5 because we can preempt the undercutter by showing *t* to be false by PR5: hence the D arguments are nested within the A arguments. We can also obtain alternative ways of showing the truth value for *r*, by typing ";":

```
[t,is,false,by,default]
[q,is,true,because,p,is,true,pr2,                      A2
   but,can,be,undercut,if,t,is,true]
[r,is,true,because,both,p,and,q,are,true,sr1]          A3
R = t ;
[r,is,false,by,default]
R = f.
```

These additional answers show two things: that we are under no obligation to show the argument against *t*, since we could presuppose it, and that there is no argument to show *r* to be false, other than the default. Our argument for *r* is not, however, entirely secure because we have a possible rebutter of PR4. We therefore see how we could establish the truth of *t*:

```
1 ?- t(1).
[s,is,true,because,assertion]                                  B1
[t,is,presumptively,true,because,s,is,true,pr5,
                                  possible,rebuttal,from,pr4]  B2
```

In ANGELIC this fails because PR4 is preferred to PR5: preferences are between *rules*. In ASPIC+, however, preferences are between *arguments*, and so the relevant preference in Figure 5 is between arguments B2 and D4. ASPIC+ proposes two principles for assessing the strength of arguments in [13]: *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 argument 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 shown above, 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 all shown, so that the weakest link can be identified. In [13] 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 [7]), and that strict rules can be considered preferred (statute law is preferred to case law). Indeed, in [13] 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.

Examination of the arguments produced by the program shows that they both ultimately depend on preferences between assertions, s itself, or u, which is used to show $\neg s$ (argument C3 in Figure 9). In [13] s is preferred to u and so argument B2 is not undermined. Note that the preference for s does not mean that we can conclude that $\neg v$ and $\neg u$ by contraposing SR7 and PR7, since defeasible rules cannot be contraposed. Similarly, even though p is an axiom we cannot contrapose the defeasible PR8 to argue that $\neg s$. ANGELIC does not enforce closure under contraposition, as recommended in [13], 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) an q(1)* to N5 and *q(0) if r(0) and p(1)* to N2.

The program is intended for use to support someone preparing to argue a case, whether dialogicaly, as in court or a debate, or monologically, as when preparing an argument for a paper or a blog. If one wishes to argue that r, one can run the query $r(R)$ as above. This will provide reassurance that one's adversary has no positive argument for $\neg r$, alerts one to the possible undercut if t is shown true, and offers a choice of how to handle this. One can either rely on the presupposition that t is false and use the default, with the potential counter punch of D1-4 if the adversary does advance B-B2. Alternatively one can preempt this move by including D1 and D4 as a nested argument in the original presentation. Either way, one will be alerted to the possible rebuttal from PR5. This can be explored by running the query $t(1)$ also shown above, revealing that, if the weakest-link principle is preferred, s may be be preferred to u and establish t. Thus one will need to be be prepared to argue for last-link, or to prefer u to s. Unfortunately neither ASPIC+ nor ANGELIC provides a means of arguing for preferences: the ordering is given for ASPIC+ and implicit in ANGELIC. To argue for preferences one needs to argue at the meta-level [12]. The example here is abstract: in a concrete case the rules could be associated with their backing, and so could be used, in the legal domain, to retrieve relevant statutes, cases and commentaries.

6. Concluding Remarks

The main purpose of this paper was to show the relation between the output from the ANGELIC methodology [1] and ASPIC+ [13]. The aim of ANGELIC is to assist with the development of concrete 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 ([1], [3]) 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 [13] that last-link is the more appropriate for legal domains can be substantiated.
- Extension to statements with degrees of acceptability. This is now considered necessary for the legal domain: [2] and [14].
- 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, 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.

[2] 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.

[3] 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.

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

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

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

[7] T. Bench-Capon and G. Sartor. A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence*, 150(1-2):97–143, 2003.

[8] G. Brewka. Weighted abstract dialectical frameworks. In *Workshop on Argument Strength*, page 9. Ruhr University, Bochum, 2016.

[9] 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.

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

[11] T. Gordon. Oblog-2: A hybrid knowledge representation system for defeasible reasoning. In *Proceedings of the 1st International Conference on AI and Law*, pages 231–239. ACM, 1987.

[12] S. Modgil and T. Bench-Capon. Metalevel argumentation. *Journal of Logic and Computation*, 21(6):959–1003, 2010.

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

[14] A. Rigoni. Representing dimensions within the reason model of precedent. *Artificial Intelligence and Law*, 26(1):1–22, 2018.

[15] 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.

[16] S. E. Toulmin. *The uses of argument*. Cambridge University Press, 1958.

[17] L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[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.