Argumentation and Reasoned Action

Proceedings of the 1st European Conference on Argumentation,

Lisbon 2015

Volume II

Edited by

Dima Mohammed

and

Marcin Lewiński

ISBN 978-1-84890-212-1

College Publications
Scientific Director: Dov Gabbay
Managing Director: Jane Spurr

http://www.collegepublications.co.uk

Original cover design by Orchid Creative www.orchidcreative.co.uk
Printed by Lightning Source, Milton Keynes, UK

Using Abstract Dialectical Frameworks to Argue about Legal Cases

LATIFA AL-ABDULKARIM
Department of Computer Science, University of Liverpool, UK
latifak@liverpool.ac.uk

KATIE ATKINSON
Department of Computer Science, University of Liverpool, UK
katie@liverpool.ac.uk

TREVOR BENCH-CAPON
Department of Computer Science, University of Liverpool, UK
tbc@liverpool.ac.uk

Recent work has shown how to map factor hierarchies for legal reasoning into Abstract Dialectical Frameworks (ADFs), by defining acceptance conditions for each node. In this paper we model as ADFs bodies of case law from various legal domains, rewrite them as logic programs, compare the results with previous legal reasoning systems and propose improvements by increasing the scope of reasoning downwards to facts.

KEYWORDS: ADF, case based-reasoning, factors, legal reasoning

1. INTRODUCTION

A recent development in computational argumentation has been Abstract Dialectical Frameworks (ADFs) by Brewka and Woltran (2010). ADFs can be seen as a generalisation of standard Argumentation Frameworks (AFs) (Dung, 1995) in which the nodes represent statements rather than abstract arguments, and each node is associated with an acceptance condition that determines when a node is acceptable in terms of whether its children are acceptable. Thus links in AFs express only one relationship, namely defeat, but ADFs can represent a variety of attack and support relations. In consequence, whereas nodes in an AF have only the single acceptance condition that all their children are defeated, nodes in ADFs can have different acceptance conditions specifically tailored for each node. In Al-Abdulkarim *et al.* (2014) it was argued that ADFs are very suitable for representing factor based reasoning with legal cases as found in the CATO system (Aleven, 1997) and as formalised in Prakken and Sartor (1998).

The key idea in Al-Abdulkarim *et al*. (2014) is that the abstract factor hierarchy of CATO (Aleven, 1997) (an extract given in Figure 1) corresponds directly to the node and link structure of an ADF, or rather (since the links are labelled "+" or "−") a Prioritised ADF (PADF) Brewka *et al.* (2013) which partitions links into supporting and attacking links, and so corresponds to the labels on the links in the factor hierarchy. To express CATO's factor hierarchy as an ADF, acceptance conditions need to be supplied for each of the nodes. Finally the logical model of IBP (Brüninghaus & Ashley, 2003) can be used to tie the various parts of the factor hierarchy together to supply decisions for particular cases. In Al-Abdulkarim *et al.* (2014) it was suggested that the acceptance conditions could be expressed as Prolog procedures. These could then be used directly to form a Prolog program that could be executed to classify cases (as to which side they are decided for) represented as sets of factors.

In this paper we will evaluate this approach. We will use US Trade secrets as the domain, allowing us to use the analysis of CATO which will permit direct comparison with CATO, IBP, the various systems used as comparators in Brüninghaus & Ashley (2003) and the AGATHA system of Chorley and Bench-Capon (2005). We will firstly consider a quantitative analysis, in terms of performance and how easily the program can be refined to improve performance, and then consider the system in terms of the transparency of its outputs, the relation to case decision texts, and the relation to formal frameworks for structured argumentation such as Prakken (2010). Finally we will consider whether the method can be readily applied to other domains, by briefly describing an application of the ADF approach to *Popov v Hayashi* and related cases as modelled in Bench-Capon (2012).

2. BACKGROUND

In this section we will recapitulate the essentials of ADFs (Brewka *et al.*, 2013), CATO (Aleven, 1997) and IBP (Brüninghaus & Ashley, 2003).

2.1 *Abstract Dialectical Frameworks*

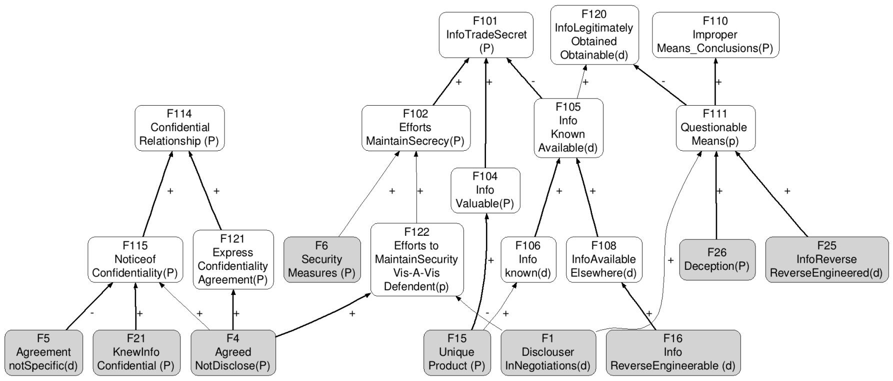
An ADF is defined in Brewka *et al*. (2013), as:

> Definition 1: An ADF is a tuple $ADF = < S,L,C >$ where S is the set of statements (positions, nodes), L is a subset of $S \times S$, a set of links, and $C = \{C_{s \in S}\}$ is a set of total functions $C_s : 2^{par(s)} \rightarrow \{t, f\}$, one for each statement s. C_s is called the acceptance condition of s.

In a Prioritised ADF, L is partitioned into L+ and L−, supporting and attacking links, respectively. Although the acceptance conditions are often expressed as propositional functions, this need not be the case: all that is required is the specification of conditions for the acceptance or rejection of a node in terms of the acceptance or rejection of its children.

2.2 *CATO*

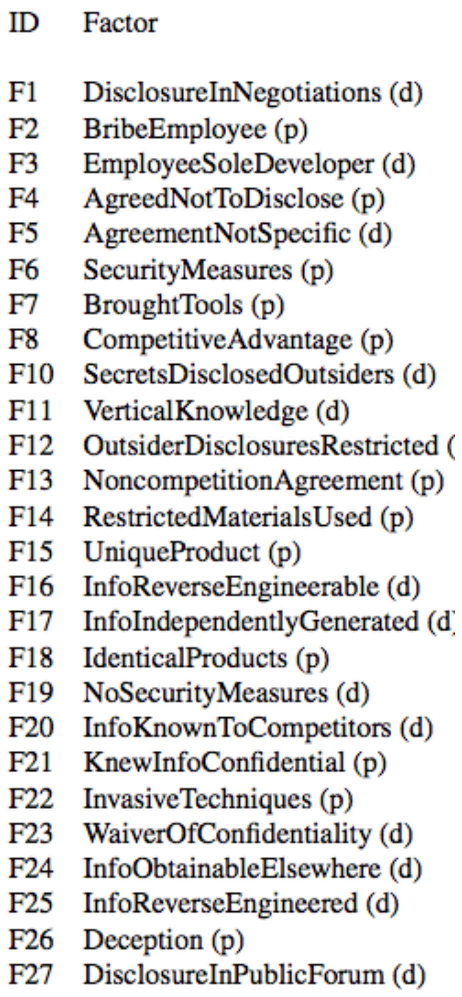
CATO (Aleven, 1997), which was developed from Rissland and Ashley's HYPO (1990), takes as its domain US Trade Secret Law. CATO was primarily directed at law students, and was intended to help them form better case based arguments, in particular to improve their skills in distinguishing cases, and emphasising and downplaying distinctions. A core idea was to describe cases in terms of factors, legally significant abstractions of patterns of facts found in the cases, and to build these base-level factors into an hierarchy of increasing abstraction, moving upwards through intermediate concerns (abstract factors) to issues. An extract from the factor hierarchy is shown in Figure 1.



F101 InfoTradeSecret (P)
F120 InfoLegitimately Obtained Obtainable(d)
F110 Improper Means_Conclusions(P)
F114 Confidential Relationship (P)
F102 Efforts MaintainSecrecy(P)
F105 Info Known Available(d)
F111 Questionable Means(p)
F104 Info Valuable(P)
F115 Noticeof Confidentiality(P)
F121 Express Confidentiality Agreement(P)
F6 Security Measures (P)
F122 Efforts to MaintainSecurity Vis-A-Vis Defendent(p)
F106 Info known(d)
F108 InfoAvailable Elsewhere(d)
F26 Deception(P)
F25 InfoReverse ReverseEngineered(d)
F5 Agreement notSpecific(d)
F21 KnewInfo Confidential (P)
F4 Agreed NotDisclose(P)
F15 Unique Product (P)
F1 Disclouser InNegotiations(d)
F16 Info ReverseEngineerable (d)
+
-

Figure 1 – CATO Abstract Factor Hierarchy from Aleven(1997)

Each factor favours either the plaintiff or the defendant. The program matches precedent cases with a current case to produce arguments in three plies: first a precedent with factors in common with the case under consideration is cited, suggesting a finding for one side. Then the other side cites precedents with factors in common with the current case but a decision for the other side as counter examples, and distinguishes the cited precedent by pointing to factors not shared by the precedent and current case. Finally the original side rebuts by downplaying distinctions, citing cases to prove that weaknesses are not fatal and distinguishing counter examples. CATO used twenty-six base level factors (there is no F9), as shown in Table 1.



ID	Factor
F1	DisclosureInNegotiations (d)
F2	BribeEmployee (p)
F3	EmployeeSoleDeveloper (d)
F4	AgreedNotToDisclose (p)
F5	AgreementNotSpecific (d)
F6	SecurityMeasures (p)
F7	BroughtTools (p)
F8	CompetitiveAdvantage (p)
F10	SecretsDisclosedOutsiders (d)
F11	VerticalKnowledge (d)
F12	OutsiderDisclosuresRestricted (
F13	NoncompetitionAgreement (p)
F14	RestrictedMaterialsUsed (p)
F15	UniqueProduct (p)
F16	InfoReverseEngineerable (d)
F17	InfoIndependentlyGenerated (d
F18	IdenticalProducts (p)
F19	NoSecurityMeasures (d)
F20	InfoKnownToCompetitors (d)
F21	KnewInfoConfidential (p)
F22	InvasiveTechniques (p)
F23	WaiverOfConfidentiality (d)
F24	InfoObtainableElsewhere (d)
F25	InfoReverseEngineered (d)
F26	Deception (p)
F27	DisclosureInPublicForum (d)

Table 1-Base Level Factors in CAT**O**

There is, however, no single root for the factor hierarchy as presented in Aleven (1997): rather we have a collection of hierarchies, each relating to a specific issue. To tie them together we turn to the Issue Based Prediction (IBP) system of Bruninghaus and Ashley (2003).

2.3 *Issue Based Prediction*

In IBP, which is firmly based on CATO, the aim is not simply to present arguments, but to predict the outcomes of cases (find for plaintiff/defendant). To enable this, the issues of CATO's hierarchy are tied together using a logical model derived from the Uniform Trade Secret Act and the Restatement of Torts. The model is shown in Figure 2.

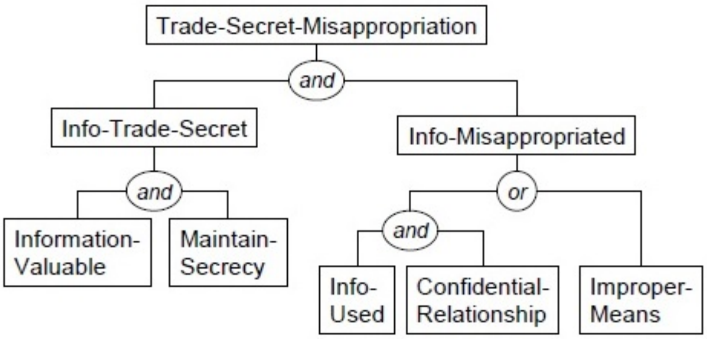


Figure 2 – IBP Logical Model from Brüninghaus & Ashley (2003)

Now consider the factor hierarchy, part of which is shown in Figure 1. We can now regard this as an ADF by forming the set S from the issues, intermediate concerns and base level factors, L+ from the links labelled "+" and L− from the links labelled "−". Using the complete factor hierarchy given in Figures 3.2 and 3.3 of (Aleven, 1997) we will have an ADF which has as its leaf nodes the base level factors of CATO. This is described in tabular form in Table 2.

ID	*S*	*L+*	*L−*
F102	EffortstoMaintainSecrecy	F6, F122, F123	F19, F23, F27
F104	InfoValuable	F8, F15	F105
F105	InfoKnownOrAvailable	F106, F108	
F106	InfoKnown	F20, F27	F15, F123
F108	InfoAvailableElsewhere	F16, F24	
F110	ImproperMeans	F111	F120
F111	QuestionableMeans	F2, F14, F22, F26	F1, F17, F25
F112	InfoUsed	F7, F8, F18	F17
F114	ConfidentialRelationship	F115, F121	
F115	NoticeOfConfidentiality	F4, F13, F14, F21	F5, F23
F120	LegitimatelyObtainable	F105	F111
F121	ConfidentialityAgreement	F4	F23
F122	MaintainSecrecyDefendant	F4	F1
F123	MaintainSecrecyOutsiders	F12	F10
F124	DefendantOwnershipRights	F3	

Table 2 CATO as ADF

The roots of CATO's hierarchies correspond to the leaves of the logical model: we can therefore form them into a single ADF by using this structure. The relevant additions to the ADF needed to integrate the IBP model are shown in Table 3 (note that F124 is not discussed in Brüninghaus & Ashley, 2003).

ID	*S*	*L+*	*L−*
F200	TradeSecretMisappropriation	F201,F203	F124
F201	Info-Miasappropriated	F110,F112,F114	
F203	Info-Trade-Secret	F102,F104	

Table 3-IBP Logical Model as ADF

IBP used 186 cases, 148 cases analysed for CATO and 38 analysed specifically for IBP. Unfortunately, these cases are not all publicly available and so we will use the 32 cases harvested from public sources by Alison Chorley and used to evaluate her AGATHA system (2005). As part of the evaluation in Brüninghaus and Ashley (2003) nine other systems were also considered to provide a comparison. Most of these were different forms of machine learning system, but programs representing CATO and HYPO were also included. IBP was the best performer: results reported in IBP (Brüninghaus & Ashley, 2003), Naive Bayes (the best performer of the ML systems), CATO, HYPO and a version of IBP which uses only the model, with no CBR component, are shown in Table 4[1].

	correct	error	abstain	accuracy
IBP	170	15	1	91.4
Naive Bayes	161	25	0	86.5
CATO	152	19	22	77.8
HYPO	127	9	50	68.3
IBP-model	99	15	38	72.6

Table 4- Results from [12]

Direct comparison with AGATHA is hampered by the fact that evaluation in AGATHA was directed towards evaluating the different heuristics and search algorithms used in that system, and so no version can be considered "definitive", and, of course, many fewer cases were used in the experiments. However, typically 27-30 of the 32 (≈ 84–93%) cases were correctly decided by the theories produced by AGATHA (Chorley & Bench-Capon, 2005).

3. ACCEPTANCE CONDITIONS

We now supply acceptance conditions for each node, to supply the elements of *C* required for the ADF. We will rely only on the definitions of the factors in Aleven (1997). We do not use precedents at this stage; as Aleven remarks:

> for certain conflicts, it is self evident how they should be resolved. For example, the fact that plaintiff's product was unique in the market (F15) arguably supports a conclusion that plaintiff's information (which is used to make the product) is not known outside plaintiff's business (F106), but not if it is also known, for example, that plaintiff disclosed its information in a public forum (F27). Common sense dictates that in those circumstances, the information is known outside plaintiff's business. It is not necessary to look to past cases to support that point. CATO's use of link strength enables a knowledge engineer to encode inferences like this. (p. 47).

From Tables 2 and 3 we can see that we have eighteen nodes to provide with acceptance conditions. One (F124) has only a single supporting

[1] No explanation for using a different number of cases for CATO and IBP model is given in Brüninghaus and Ashley (2003).

child: thus the acceptance condition will be Parent ←→ Child. We will write this (and the other acceptance conditions) as a set of tests for acceptance and rejection, to be applied in the order given, which allows us to express priority between them. The last test will always be a default. We choose this form of expression because we find it easier to read in many cases, because it corresponds directly to the defeasible rules with priorities used in formalisms such as ASPIC+ (Prakken,2010), and because it is directly usable as Prolog code. Thus we write Parent ←→ Child as

> Accept Parent if Child.
> Reject Parent.

Where NOT is required we use negation as failure. The tests are individually sufficient and collectively necessary, ensuring equivalence with the logical expression (see Clark, 1978). Six nodes (F201, F203, F105, F108, F114 and F124) have only supporting links: these can be straightforwardly represented using AND and OR. We followed the IBP model for the two nodes taken from that model (F201 and F203), and used OR for the other four. The most complicated was InfoMisappropriated (F201):

> Accept InfoMisappropriated if F114 AND F112.
> Accept InfoMisappropriated if F110.
> Reject InfoMisappropriated.

Five nodes have one supporting and one attacking link. These are best seen as forming an exception structure: accept (reject) the parent if and only if supporting (attacking) child unless attacking (supporting) child. Note that the exception may be the supporting or the attacking child: in the former case the default will be reject, and in the latter the default will be accept. Thus:

> Accept Parent if Support AND (NOT Attack).
> Reject Parent.
> Reject Parent if Attack AND (NOT Support).
> Accept Parent.

For F110, F120 and F121 the attacking child is the exception, while for F122 and F123 the supporting links are the exceptions. This leaves seven nodes. For F200 we regard the attacking link as an exception to the case where the conjunction of the supporting links holds.

For F104 and F112 we see the supporting links as offering disjoint ways of accepting the parent, and the attacking child as a way of establishing that the factor is not present. We default to yes because in many cases there are no factors for either side present relating to this point. We take it that this factor was often simply accepted on the facts and uncontested, and so there was no discussion on the point. A full description of all the truth conditions is given in Al-Abdulkarim *et al.* (2015).

4. PROLOG PROGRAM

The Prolog[2] program was formed by ascending the ADF, rewriting the acceptance conditions as groups of Prolog clauses to determine the acceptability of each node in terms of its children. The tests were restated using the appropriate syntax, with some reporting to indicate whether the node is satisfied (defaults are indicated by the use of "accepted that"), and some control to call the procedure to determine the next node, and to maintain a list of accepted factors. Examples can be found in Al-Abdulkarim *et al*. (2015).

Each of the tests in the acceptance condition is applied in a separate clause, using the set of factors currently identified as present in the case, before proceeding to the next factor, with the current factor added to the applicable factors if it is accepted. To allow completion of the database (Clark, 1978), a final clause is added to catch any case not covered by any of the preceding clauses. These defaults may favour either side. In some cases, the default is accept because few case descriptions related to these abstract factors, although they are a *sine qua non* for any claim. Our belief is that these aspects were uncontested and so the factors were not explicitly discussed in the trial, and so do not appear in the CATO analysis. Where we felt it was clear that the factor needed to be explicitly established, the default was reject.

The above demonstrates that it is a straightforward and reasonably objective process to transform a factor based analysis such as is found in (Aleven, 1997) to an executable program via an ADF. Although judgement was sometimes required to form the acceptance conditions, we would suggest that such judgements were not difficult to make. Moreover if there are difficult choices, the effect of the alternatives can be compared on a set of test cases. Overall the relatively small number of factors relevant to particular nodes greatly simplifies the task.

[2] Prolog was used because of its closeness to the acceptance conditions, and made the implementation quick, easy and transparent.

5. RESULTS

We can now run the program on the cases. We represent the cases as a list of base-level factors. For example, the Boeing case[3] is represented as case(boeing,[f4,f6,f12,f14,f21,f1,f10]).

```
giving output:
1 ?- go(boeing).
accepted that defendant is not owner of secret
efforts made vis a vis outsiders
efforts made vis a vis defendant
there was a confidentiality agreement
defendant was on notice of confidentiality
 there was a confidential relationship
accepted that the information was used
questionable means were used
accepted that the information was not available elsewhere
accepted that information is not known
accepted that the information was neither known nor
available
accepted that the information was valuable
not accepted that the information was legitimately obtained
improper means were used
efforts were taken to maintain secrecy
information was a trade secret
a trade secret was misappropriated
find for plaintiff
boeing[f200, f201, f203, f102,f110,f104,f111,
f112,f114,f115,f121,f122,f123, f4,f6,f12,f14,f21,f1,f10]
decision is correct
```

The initial program correctly classified 25 out of the 32 cases (78.1%). While all ten of the cases won by the defendant were correctly classified, seven of the 22 cases won by the plaintiff were not. The figure for correct answers is remarkably close to the 77.8% reported for the version of CATO used in (Brüninghaus and Ashley, 2003), which, of course, uses exactly the same analysis of the domain and cases that we have adopted here. Thus as a first conclusion we can tentatively suggest that executing the analysis in Aleven (1997) as an ADF produces very similar results to those obtainable using the original CATO program (albeit we are using a smaller set of cases). We can now investigate how the initial program might be improved.

[3] The Boeing Company v. Sierracin Corporation, 108 Wash.2d 38, 738 P.2d 665 (1987).

The wrongly predicted cases were:

```
case(spaceAero,[f8,f15,f18,f1,f19]).
case(televation, [f6,f12,f15,f18,f21,f10,f16]).
case(goldberg,[f1,f10,f21, f27]).
case(kg,[f6,f14,f15,f18,f21,f16,f25]).
case(mason,[f6,f15,f21,f1,f16]).
case(mineralDeposits,[f18,f1,f16,f25]).
case(technicon,[f6,f12,f14,f21,f10,f16,f25]).
```

Examination of the cases showed that five of the seven had F16 (ReverseEngineerable) present and that these cases were the only cases found for the plaintiff with F16 present. The problem in these five cases is that the program finds for the defendant because the information is available elsewhere (F105). This is established by the presence of ReverseEngineerable and is unchallengeable. Examination of the ADF shows that F16 is immediately decisive: if that factor is present, there is no way the plaintiff can demonstrate that the information is a trade secret. Goldberg[4] also fails through F105 (information known or available), since disclosure in a public forum (F27) is sufficient to deny the information trade secret status. It would appear that we could significantly improve performance by refining this branch to allow the plaintiff some way to defend against, in particular, F16. See Al-Abdulkarim *et al*. (2015) for discussion of the texts of the decisions and possible refinements. The refined program can equal or better the performance of any of the existing systems.

5.1 Discussion

By using the ADF we can readily explain the points at which the acceptance conditions do not concur with the decisions taken in the actual cases. We can then return to the original decisions and use them to determine possible refinements to the representation. In some cases, the problem seems to lie with the attribution of the factors. Such matters were contested in the actual case, and ascribing the presence or absence of particular factors requires interpretation of the case by the analyst. The interpretation cannot be disputed without descending to the level of facts as advocated by Atkinson *et al.* (2013) and Al-Abdulkarim *et al.* (2014). Addition of the fact layer has been the subject of work subsequent to that reported here. Other decisions suggest that we may wish to modify the description of factors intended to guide the

[4] Goldberg v. Medtronic, 686 F.2d 1219 (7th Cir. 1982).

analyst. Adding or removing a factor to or from a particular case provides a local solution which will solve a problem with a particular case. Our results, however, indicated a general problem which was applicable to several cases: the dominant affect of F16, reverse engineerable. It seemed clear to us that the presence of F16 should not by itself be sufficient for a finding for the defendant. Again the decisions themselves suggested several possible ways of arguing against F16: in particular the use of restricted materials and the uniqueness of the product. Either or both of these exceptions could be incorporated in the ADF without adversely affecting any of the test cases, but we would need to have a reasonably large set of new cases in order to evaluate the different solutions and to guard against over fitting.

Finally it should be conceded that the decisions themselves may be erroneous. Assuming that there are least some poor decisions which we would not wish to serve as precedents, we should be willing to tolerate a certain number of divergences from our results.

To summarise:

- Simply translating the analysis of (Aleven, 1997) into an ADF and executing the resulting program gave results almost identical to those found for CATO in the IBP experiments reported in Brüninghaus and Ashley (2003). Note that this is achieved without need for balancing of pro and con factors central to existing case based reasoning systems.
- The reasons for the “incorrect” decisions can be readily identified from the output and the ADF, as we saw from the discussion of the wrongly decided cases above.
- Examination of the texts of the decisions readily explained why the results diverged, and suggested ways in which the analysis could be improved, either at the case level by changing the factors attributed, or at the domain level by including additional supporting or attacking links.

From this we conclude that use of ADFs provides good performance, and has a number of positive features from a software engineering (and domain analysis) standpoint, which would enable the ADF to be refined and performance improved. We also believe that we do need to include a fact layer to permit increased transparency in the ascription of factors to cases.

6. QUALITY OF EXPLANATIONS

As the Prolog program proceeds it reports on the acceptability or otherwise of the various abstract factors and the resolution of issues. As shown above, this provides an excellent diagnostic for divergent decisions, but how does it measure up the actual decisions found in cases?

Of course, without facts, we will not be able to follow the decision very closely. But consider a reordering of the elements of our decision for, say Boeing. We also omit some elements, and add a little linking text. Recall too that we wrote the program used thus far to “decide” the cases: in a version to supply explanation we would want to customise the text reports to indicate the particular clause being used for a node by giving the base level factors used. Below is what a decision might look like: we show the current program output in boldface, possible clause-specific customisations in italics and linking text in ordinary font.

> We **find for plaintiff**. The **information was a trade secret**: **efforts were taken to maintain secrecy**, *since disclosures to outsiders were restricted and the defendant entered into a non-disclosure agreement and other security measures were applied.* The **information was unique**. It is **accepted that the information was valuable** and it is **accepted that the information was neither known nor available**.
> **A trade secret was misappropriated**: **there was a confidential relationship** *since the defendant entered into a non-disclosure agreement* and it is **accepted that the information was used.**
> Moreover **improper means were used** *since the defendant used restricted materials.*

This seems to have the makings of a reasonable explanation. There are two problems: it does not indicate what the defendant contended, since the clauses of the program which were not reached do not feature in the report, and, of course, the facts on which the finding are based are not present. None the less, we find the output a distinct improvement on previous work such as Chorley and Bench-Capon (2005). We believe that the output from the current program could be readily used to drive a program of the sort envisaged by Branting (1993), and that this will become even more useful when we have added a fact layer to allow the explanation of the attribution of factors.

7. APPLICATION TO A SECOND DOMAIN

In the above we have considered the approach with respect to a single domain. If the approach is to be of general significance, however, it needs to be applicable to other domains. This section describes a further exercise designed to show that the approach is more generally applicable. We will apply the method to a domain which has often been used as an illustration of factor based reasoning: the wild animals cases and *Popov v Hayashi*. The wild animals cases were introduced into AI and Law in Berman and Hafner (1993) and extended to the baseball case of Popov in Wyner *et al*. (2007). We will use the factor-based analysis of Bench-Capon (2012) as our starting point.

Briefly the wild animals cases concern plaintiffs chasing wild animals when their pursuit was interrupted by the defendant. Post was chasing a fox for sport. Keeble was hunting ducks, Young fish and Ghen a whale, all in pursuit of their livelihoods. *Popov v Hayashi* concerned disputed ownership of a baseball (valuable because it had been hit by Barry Bonds to break a home run record). Popov had almost completed his catch when he was assaulted by a mob of fellow spectators and Hayashi (who had not taken part in the assault) ended up with the baseball when it came free. The wild animals cases were cited when considering whether Popov's efforts had given him possession of the ball.

Thirteen, base-level, factors are identified in Bench-Capon (2012). The first task is to form them (together with appropriate abstract factors) into a factor hierarchy, to use as the node and link structure of our ADF. This factor hierarchy is shown in Figure 3: some adaptations have been made; for example, we include a factor Res (Residence Status) to indicate the attachment of the animals to the land, since it appears to make a difference whether they are there permanently, seasonally, habitually, occasionally, or whatever. The nodes and links are given in Table 5.

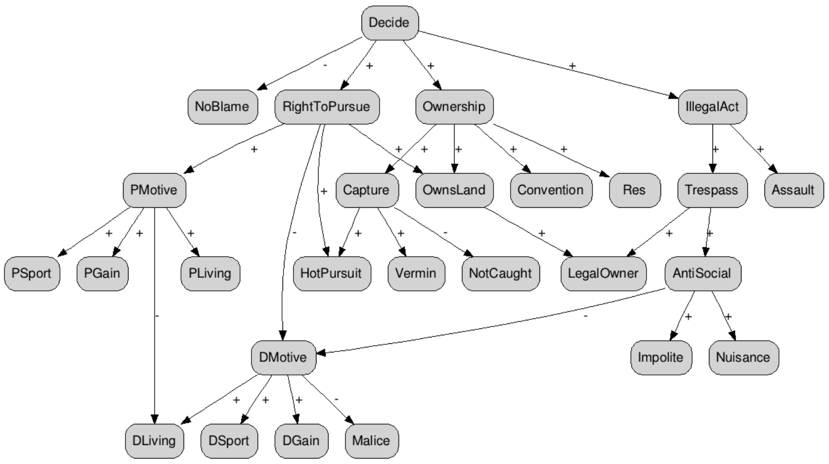


Figure 3-Factor Hierarchy/ADF for Popov

S	L+	L-
Decide	Ownership,RightToPursue,IllegalAct	NoBlame
Capture	HotPursuit, Vermin	NotCaught
Ownership	Convention, Capture, OwnsLand, Res	
PMotive	Pliving, PSport, PGain	DLiving
DMotive	DLiving, DSport, DGain	Malice
OwnsLand	LegalOwner	
RightToPursue	OwnsLand, Pmotive, HotPursuit	DMotive
AntiSocial	Nuisance, Impolite	DMotive
Trespass	LegalOwner, AntiSocial	
IllegalAct	Assault, Trespass	

Table 5-Popov as ADF

We now supply acceptance conditions for the nine non-leaf nodes.

1. Decide for Plaintiff if NOT (NoBlame) AND (Ownership OR (RightToPursue AND IllegalAct))
2. Ownership if (OwnsLand AND Resident) OR Convention OR Capture
3. Capture if NOT (NotCaught) OR (Vermin and HotPursuit)
4. RightToPursue if OwnsLand OR (HotPursuit AND PMotive AND (NOT (better) DMotive))
5. PMotive if PLiving OR (PSport OR PGain) AND (NOT DLiving)
6. DMotive if NOT Malice AND (DLiving OR DSport OR DGain)
7. IllegalAct if Trespass OR Assault
8. Trespass if LegalOwner AND AntiSocial
9. AntiSocial if (Nuisance OR Impolite) AND (NOT DLiving)

The only real controversy here is with the determination of Right to Pursue when both the plaintiff and the defendant have good motives. Essentially we want to say that if the land is not owned by one of them, the right to pursue is given to the party with the better motive. The remainder seem fairly uncontroversial. The acceptance conditions can easily be expressed as Prolog procedures and then embedded in code as was done for CATO. We can now execute the program. Running the case for Young v Hitchens produces the output (note that the program abbreviates factor names):

```
1 ?- go(young).
the plaintiff had not captured the quarry
the plaintiff did not own the quarry
plaitiff has good motive
defendant has good motive
```

```
plainiff did not own the land
plainiff had a right to pursue the quarry
defendant committed no antisocial acts
defendant committed no trespass
no illegal act was committed
do not find for the plaintiff
find for the defendant
young[rtToPursue,dMotive,pMotive,nc,hp,imp,pliv,dliv]
```

We produce correct results from all five cases discussed in Bench-Capon (2012), and on this basis we believe that the ADF representation can be used to encapsulate the knowledge of the domain. We cannot evaluate it as a decision making program since there are insufficient cases available, but this suggests that the method can be applied straightforwardly to a second domain to construct an executable program. In general we believe that the method can be applied to any domain for which factor based reasoning in the CATO (or HYPO or IBP) style is appropriate. This has encouraged us sufficiently to attempt to apply the method to a larger scale problem in the domain of the US automobile exception to the fourth amendment rule for which there is no accepted analysis into factors available, so we that need to start from the case decision texts: we will also incorporate a fact layer in this domain. This is the subject of the next stage of our project.

8. CONCLUDING REMARKS

In this paper we have evaluated an approach to reasoning with legal cases described in terms of factors using Abstract Dialectical Frameworks, as described and advocated in Al-Abdulkarim *et al.* (2014). We find that:

- The success of the implementation depends to a large extent on the quality of the analysis. Making a direct translation of the analysis of CATO (Aleven, 1997) yields a success rate almost identical to that found for CATO in (Brüninghaus & Ashley, 2003). This is a creditable 78.1% of the cases decided "correctly".
- The ADF does, however, provide very transparent output that identifies precisely where the outcomes suggested by the implementation diverge from the actual outcomes. Now reading the original decision texts suggests one of four solutions. These are, in ascending order of divergence from the original analysis:
  1. Removing a factor wrongly attributed to the case
  2. Adding a factor wrongly omitted from the case
  3. Modifying an acceptance condition: e.g. changing the priorities

4. Modifying the ADF: e.g. adding a supporting or attacking node for the problem node. Often several of these modifications can potentially solve the problem, and the choice is made according to the context provided by the other divergent cases we are trying to accommodate.

- The ADF approach provides a good way of using a set of test cases to refine an initial analysis.
- The output from the program provided good diagnostics and a reasonable explanation of the outcome. Our output does, however, currently lack the citations and facts which are prominent in actual decisions.
- The method emphasises reasoning with portions of precedents, rather than whole cases. We believe that this does correspond to legal practice as manifest in real decisions.
- The method can be applied to different domains. We believe that any domain for which factor based reasoning would be appropriate would be amenable to this method.

We find all of this encouraging. The next important step will be to extend the method to the fact level, so as to permit argument about the ascription of factors, and to be able to ground our explanations in the particular facts of a case. Once the method has been extended to include the facts of particular cases at the lowest level of the ADF, a program to present the output in a form resembling the texts of actual decisions can also be considered.

REFERENCES

Al-Abdulkarim, L., Atkinson, K., & Bench-Capon, T. (2014). Abstract dialectical frameworks for legal reasoning. *Proceedings of Jurix 2014*, 61–70.

Al-Abdulkarim, L., Atkinson, K., & Bench-Capon, T. (2015). Evaluating the Use of Abstract Dialectical Frameworks to Represent Case Law. *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law*, 156–160.

Aleven, V. (1997). *Teaching case-based argumentation through a model and examples*. PhD thesis, University of Pittsburgh.

Ashley, K. (1990). *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge, MA. : MIT Press.

Atkinson, K., Bench-Capon, T., Prakken, H., & Wyner, A. (2013). Argumentation schemes for reasoning about factors with dimensions. *Proceedings of JURIX 2013*, 39–48.

Bench-Capon, T. (2012). Representing Popov v Hayashi with dimensions and factors. *Artificial Intelligence and Law*, *20*(1), 15–35.

Berman, D., & Hafner, C. (1993). Representing teleological structure in case-based legal reasoning: The missing link. *Proceedings of the Fourth International Conference on Artificial intelligence and Law*, 50–59.

Branting, L. K. (1993). An issue-oriented approach to judicial document assembly. *Proceedings of the 4th ICAIL*, 228–235. ACM.

Brewka, G., Strass, H., Ellmauthaler S., Wallner, J., & Woltran S. (2013). Abstract dialectical frameworks revisited. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press.

Brewka G., & Woltran S. (2010). Abstract dialectical frameworks. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference*.

Brüninghaus, S., & Ashley, K. (2003). Predicting outcomes of case-based legal arguments. In *Proceedings of the 9th ICAIL*, 233–242.

Chorley, A., & Bench-Capon, T. (2005). Agatha: Using heuristic search to automate the construction of case law theories. *Artificial Intelligence and Law*, *13*(1), 9–51.

Clark, K. L. (1978). Negation as failure. *Logic and data bases*, 293–322. Springer.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-person games. *Artificial Intelligence*, *77*, 321–357.

Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation*, *1*(2), 93–124.

Prakken, H., & Sartor, G. (1998). Modelling reasoning with precedents in a formal dialogue game. *Artificial Intelligence and Law*, *6*(2-4), 231–287.

Wyner, A. Z., Bench-Capon, T. J. M., & Atkinson, K. (2007). Arguments, values and baseballs: Representation of popov v. hayashi. *JURIX2007*, 151–160.