Evaluating the Use of Abstract Dialectical Frameworks to Represent Case Law

Latifa Al-Abdulkarim
Dept. of Computer Science
University of Liverpool
UK
latifak@liverpool.ac.uk

Katie Atkinson
Dept. of Computer Science
University of Liverpool
UK
katie@liverpool.ac.uk

Trevor Bench-Capon
Dept. of Computer Science
University of Liverpool
UK
tbc@liverpool.ac.uk

ABSTRACT

Abstract Dialetical Frameworks (ADFs) are a recent development in computational argumentation which are, it has been suggested, a fruitful way of implementing theories of case law expressed in terms of factors. In this paper we evaluate this proposal, by representing the CATO analysis using ADFs. We evaluate the ease of implementation, the efficacy of the resulting program, ease of refinement of the program, transparency of the reasoning, relation to formal argumentation techniques, and transferability across domains[1].

1. INTRODUCTION

A recent development in computational argumentation has been Abstract Dialectical Frameworks (ADFs) [9]. ADFs can be seen as a generalisation of standard Argumentation Frameworks (AFs) [12] in which the nodes represent statements rather than abstract arguments, and each node is associated with its own acceptance condition which determines its acceptability in terms of whether its children are acceptable. Thus while links in AFs express only one relationship, namely *defeat*, ADFs can represent a variety of attack and support relations. In [1] it was argued that ADFs are very suitable for representing case law domains represented in terms of factors as in CATO [4] and as formalised in [15] and [13].

The key idea of [1] is that the abstract factor hierarchy of CATO [4] (depicted in Figures 3-2 and 3-3 of [4]) corresponds directly to the node and link structure of an ADF, and this will provide a good basis on which to construct an executable theory of the law embodied in a set of precedents. ADFs provide a complete methodology to enable a domain analysis to be realised in a formal structure about which properties can be proved. This structure can be rewritten in executable form to enable the analysis to be tested and refined. We believe that this methodology brings improvements in formal clarity and transparency of execution over existing approaches. We will evaluate this approach using US trade secrets as the domain. This allows us to use the analysis of CATO and permits comparison with CATO, IBP and the other systems evaluated in [10], and the AGATHA system [11].

[1]This paper is a revised and shortened version of [2], which contains additional detail on several points.

ICAIL '15, June 08–12, 2015, San Diego, CA, USA
ACM 978-1-4503-3522-5/15/06.
http://dx.doi.org/10.1145/2746090.2746111.

2. BACKGROUND

In this section we will recapitulate the essentials of ADFs [9], CATO [4] and IBP [10].

An ADF is defined in [9] as a tuple $ADF = < S, L, C >$ where S is a 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.

CATO [4], which was developed from Rissland and Ashley's HYPO, most fully described in [5], takes as its domain US Trade Secret Law. CATO was primarily directed at law school 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. Each factor favours either the plaintiff or the defendant. CATO 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 [4]: the list can also be found in Table 3 of [3] in this volume.

There is, however, no single root for the factor hierarchy as presented in [4]: rather there is a collection of hierarchies, each relating to a specific issue. To tie them together we turn to the Issue Based Prediction (IBP) system of Brüninghaus and Ashley [10].

In IBP, which is firmly based on CATO, the aim is not simply to present arguments, but to predict the outcomes of cases. 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*. This use of a logical model at the top level along with precedent based reasoning to determine the status of the leaf nodes was previously used in CABARET [16]. As described in [1] the factor hierarchy can be seen as an ADF by forming the set S from the issues, intermediate concerns and base level factors, $L+$ from the links labeled "+" and $L-$ from the links labeled "−".

Using the complete factor hierarchy given in Figures 3.2 and 3.3 of [4] we will have an ADF which has as its leaf nodes the base level factors of CATO. The roots of CATO's hierarchies correspond to the leaves of the IBP logical model: we can therefore tie them into a single ADF by using this structure. The resulting ADF is given in tabular form in Table 1 (nodes from IBP are 200+).

ID	*S*	*L*+	*L*−
F200	TradeSecretMisappropriation	F201,F203	F124
F201	InfoMiasappropriated	F110,F112, F114	
F203	InfoTradeSecret	F102,F104	
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 1: IBP and CATO 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 [11]. As part of the evaluation in [10], nine other systems were also considered to provide a comparison. Most of these were forms of machine learning system, but programs representing CATO and HYPO were also included. IBP was the best performer (and is still the benchmark): results reported in [10] for IBP, 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 2.

	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 2: Results from [10]

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". Also many fewer cases were used in the experiments. 27-30 of the 32 ($\approx 84-93\%$) cases were correctly decided by the theories produced by AGATHA [11], depending on the precise heuristic used to guide the search.

3. ACCEPTANCE CONDITIONS

We now complete the ADF by supplying the acceptance conditions, *C*. We will rely only on the definitions of the factors in [4], as informed by the strong and weak links identified there. We do not use precedents at this stage; as Aleven remarks: "for certain conflicts, it is self evident how they should be resolved. ... It is not necessary to look to past cases to support that point" ([4] p47).

Note that we are now expressing a theory embodying the CATO analysis as an ADF: essentially a knowledge engineering task. The executable code will apply the theory derived from the precedents by the knowledge engineer, rather than itself reasoning with the precedents directly. From Table 1 we can see that we have eighteen nodes to provide with acceptance conditions. One (F124) has only a single supporting child: thus the acceptance condition will be $Parent \longleftrightarrow Child$. 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 assign a default value. Where NOT is required we use negation as failure. The tests are individually sufficient and collectively necessary, and so satisfy the closed world assumption, ensuring equivalence with the logical expression.

Nodes which have only supporting links 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.

Nodes that have one supporting and one attacking link are best seen as forming an exception structure: accept (reject) the parent 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*.

This leaves seven nodes, each of which require individual treatment. For details of their acceptance conditions see [2].

4. PROLOG PROGRAM

The Prolog program to apply the theory represented in the ADF was formed by ascending the ADF, starting at the leaf nodes, and rewriting the acceptance conditions as groups of Prolog clauses to determine the acceptability of each node in terms of its children. This requires restating the tests using the appropriate syntax, adding some reporting to indicate whether the node is satisfied and some control to call the procedure to determine the next node, and to maintain a list of accepted factors. Examples of the Prolog code are given in [2].

Each test in the acceptance conditions 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 factor under current consideration added if it is accepted. A final clause is added to each procedure in case none of the preceding clauses applies. These defaults may favour either side. It is thus a straightforward and reasonably objective process to transform a factor based analysis such as is found in [4] into 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 should there be any difficult choices, the effect of the alternatives can be compared on a set of test cases, and the program refined if the effect is not correct. We will consider the refinement process in section 4.2. Overall the relatively small number of factors relevant to particular nodes (never more than seven, and half the nodes have only two), greatly simplifies the task. The result is a theory of the domain case law expressed formally as an ADF, and executably as a Prolog program.

4.1 Executing the Program

We can now run the program on the cases. We represent the cases as a list of base-level factors, and get as output a list of factors (base level and abstract) considered present, a partial list of absent factors, the classification according to the ADF and whether this agrees with the actual decision. The list of absent factors is partial since once the status of a node is known (e.g. a disjunct is satisfied) the remaining tests are not applied. The actual output from an example case can be found in [2].

The initial program correctly classified 25 out of the 32 cases (78.1%) of the cases. 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 [10], which, of course, uses exactly the same analyisis of the domain and cases that we have adopted here: our aim was to represent the theory embodied in the CATO analysis. Thus as a first conclusion we can tentatively suggest that executing the analysis in [4] 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.

Examination of the misclassified cases showed that five of the seven had *ReverseEngineerable* (F16) 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 *InfoAvailableElsewhere* (F108), is established by the presence of *ReverseEngineerable* and is unchallengeable. F16 is immediately decisive: if that factor is present, there is no way to argue that the information is a trade secret. A sixth case (*Goldberg v Medtronic*) also fails through *InfoAvailableElsewhere*, since *DisclosureInPublicForum* (F27) is also 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. CATO can be expected to be more robust in the face of imperfect analysis than an approach based on a logical model: because CATO generates arguments based on considering all the available factors taken together, it is less likely to have the outcome determined by a single factor.

4.2 Refinement

We refine our theory (and hence the program) by considering the decisions in the misclassified cases[2]. In *Goldberg* it is explicitly stated that "*The district court found that Medtronic could not avoid its obligation of confidence due to the availability of lawful means of obtaining the concept when those means were not employed. We affirm.*" Thus F27, the factor which was decisive for our progam, was in fact explicitly held to be insufficient in the actual decision. Whether this decision was correct or not is not for us to say, but it does explain why our program misclassified the case. We can either redefine F27 to include the defendant's actual *use* of this public domain knowledge, so that it is not present in *Goldberg*, or allow F21 as an exception to F27 in determining the acceptance of F106. Since we have no other case with F21 and F27 both present, we cannot choose between these two solutions on the precedents available to us.

We now turn to the problem created by *ReverseEngineerable*, F16, which applies if "*Plaintiff's information could be ascertained by reverse engineering, that is, by inspecting or analyzing plaintiff's product (regardless of whether defendant actually obtained the information in this way)*" [4]. There are defences against reverse engineerability. *Mason v. Jack Daniel Distillery* and *KFC v Marion Kay* suggest that the uniqueness of the product (F15) might be a factor capable of attacking the acceptability not only of F106 (as identified in CATO) but of F108 as well. Adding F15 as an exception to F16 would give the correct decision in *Televation*, *KG* and *Mason*.

Moreover in *Televation* the phrase "readily ascertainable" is used: suggesting that the ease of reverse engineering the product needs to be considered. In two of the cases (*KG* and *Technicon*) restricted materials were used by the defendants (F14), strongly implying that the information was not, in fact, readily ascertainable. Moreover the decision in *Mineral Deposits*, strongly suggests to us that F14 was also in fact present in this case. In *Televation*, whether the secret counted as reverse engineerable was contested: and there is a strong suggestion that the court in fact believed that copies of the plaintiff's drawings had, in fact been used by the defendant, which would mean that F14 would apply there also.

The decisions thus suggest several exceptions to F16 as a support for F108: especially uniqueness of the product and use of restricted materials. Incorporating those exceptions would raise success of our program to 29 out of 32 (90.6%), and removing F27 from *Goldberg* (or allowing F21 as an exception) and adding F14 to *Mineral Deposits*, both of which seem eminently justifiable from the decisions in the cases concerned, would also correctly classify these cases (96.8%).

This leaves only *Space Aero* as an unexplained failure. It can be seen from the output for this case (given in [2]) that it fails on two branches: the information is not a trade secret because no security measures were taken, and because it appears that no confidential relationship existed. A key feature of this case is that the defendants were former employees of the plaintiff, and had been provided with the disputed information when employed by the plaintiff because they needed it to carry out their duties. The decision itself states "*The testimony, taken as a whole, convinces us that Darling took precautions to guard the secrecy of its process which, under the circumstances, were reasonably sufficient*." This suggests to us that F19 (*NoSecurityMeasures*) was not, in fact, accepted by the court as present: removing this factor from the case is enough to establish that there was a trade secret. On the issue of confidentiality, we read that although there was no formal non-disclosure agreement, by removing certain drawings without the plaintiff's knowledge the employees had "*violated the duty of fidelity and trust which they owed to*" their employer. Again we cannot comment on whether this decision was correctly made or not, but it does seem that at least F21 (*KnewInfoConfidential*) should be included. If this is added, the program establishes a confidential relationship and finds for the plaintiff.

4.3 Discussion

The previous section shows that by using the ADF we can readily identify the points at which the acceptability conditions do not yield 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. Should *Goldberg* really have F27? Should *Mineral Deposits* have F14? Should *Space Aero* include F14 or F21 and exclude F19? 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 e.g. [6] and [1]. As well as disputed factors, a decision like *Goldberg* suggests that we may wish to modify the description of factors intended to guide the analyst. In that decision it was suggested that to count for the defendant, the information not

[2]A more detailed account of our use of the decisions is given in [2].

only had to be publicly available but that the public source needed to be known to and *used by the defendant*, which would narrow the applicability of F27 compared with [4].

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 influence of F16, reverse engineerable. It seemed clear to us on reading the decisions 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 (F14) or the uniqueness of the product (F15). Either or both of these exceptions could be incorporated in the ADF without adversely affecting any currently available 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 [4] into an ADF and executing the resulting program gave results almost identical to those for CATO in the IBP experiments [10]. This is achieved without need for the balancing of pro and con factors central to existing case based reasoning systems, giving greater transparency.

- 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 expressing the analysis as an ADF can provide good performance, and has a number of positive features from a software engineering (and domain analysis) standpoint, which enables the ADF to be refined and performance improved. We also believe that it suggests the need to include a fact layer to permit increased transparency in the ascription of factors to cases.

5. OTHER OBSERVATIONS

In this section we discuss several other points raised by the evaluation.

5.1 Reporting the Decisions

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 to the actual decisions found in cases?

While the output does offer a reasonable explanation of the decision, of the sort that might be expected from a rule-based expert system, there are three problems when considered as the kind of reasoned justification of the decision that a lawyer might expect. It does not state the facts on the basis of which the factors were attributed; it does not indicate the precedents from which the factors and their acceptance conditions were derived, and it does not indicate what the losing side contended, since the clauses of the program which were not reached in the program do not feature in the explanation, None the less, we find the output a distinct improvement on previous work such as [11], since it does present a clear line of reasoning. In order to remedy these problems, we would need to extend the program to incorporate a layer of facts, supply each clause with the precedents which justify it, and alter the control to explore all lines of reasoning, so the rejected contentions can be included.

5.2 Portions of Precedents

Using the ADF approach, we do not see confrontations between large sets of pro and con factors covering the whole case. Instead factors are opposed to one another in the context of accepting or rejecting particular nodes, and so will represent a specific point in the argument. Thus two cases may be identical with respect to the factors determining whether a relationship was confidential, while very different with respect to whether the information was a trade secret. Two points are significant here: first that some apparent distinctions are insignificant, since they relate to different issues: this was partly what the factor hierarchy was introduced in CATO to address. More importantly, since particular acceptance conditions can be traced back to subsets of factors present in particular precedents, a precedent might be citable to establish that, for example, a confidential relationship existed, (favouring the plaintiff on that issue) even though the case as a whole was found for the defendant, because of some other issue (such as that the information was not, in fact, a trade secret). For this reason the ADF can be seen as embodying reasoning with portions of precedent, as urged in [8].

The use of portions of precedents, rather than requiring cases to be considered only as the complete set of factors they contain is supported by a reading of the decisions in the various cases: rarely do they begin by citing a precedent and then discuss similarities and differences. Instead they use precedents at particular points of the decisions to identify questions and issues to be addressed, and to justify answers and consequences. This is effectively what is done in the ADF approach: competing factors are considered in the context of accepting or rejecting particular nodes.

Another reason to consider portions of precedents is provided by [13]. In that paper *a fortiori* reasoning was explained in terms of preferring a rule with a *subset* of the factors available for the winning side to the rule using *all* the factors available to the losing side. That paper gave, however, no indication of how this subset should be chosen. The output from the ADF in contrast, does show which factors were instrumental and active in winning the case.

5.3 Relation to Structured Argumentation

The overall output of our system bears a strong resemblance to the kind of structured argumentation found in ASPIC+ [14]. The union of the acceptance conditions can be seen as the underlying knowledge base. Determining the acceptance or rejection of the various nodes produces sub-arguments, and these can be linked to produce the argument for finding for the plaintiff (or defendant) which follows the argument-subargument structure of ASPIC+ arguments. There are also differences: because the ordering of clauses expresses priority between arguments only the winning arguments are generated. Thus the output does not include all arguments, but only the winning line of argument. Where, however, potential attackers are children rather than siblings, but are not acceptable, this is reported. Thus although there are correspondences, especially through the argument-subargument structure, the control regime employed by the program means that there are also impor-

tant differences. These relate mainly to conflicts: the output shows only the winning side of the case, and does not provide a good record of the rejected arguments available to the losing side. None the less, the correspondences are such that this is worth exploring further, using a different control regime in the implementation to generate all arguments, as a potentially fruitful way of formalising the reasoning.

5.4 Applicability 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. We have applied the method to another domain, one which has often been used as an illustration of factor-based reasoning chosen because analyses are available: the wild animals cases and *Popov v Hayashi*. We used the factor-based analysis of [7] as our starting point.

Our ADF representing the the factor hierarchy, acceptance conditions and output for *Young v Hitchens* are all given in [2].

The program produced correct results from all five cases discussed in [7], suggesting that the ADF representation can be used to encapsulate the knowledge of the domain as represented in [7]. As such we believe that any analysis of a case law domain in terms of factors can be readily represented as an ADF.

6. CONCLUDING REMARKS

In this paper we have evaluated an approach to representing a case law theory described in terms of factors using ADFs, as described and advocated in [1]. We drew a number of conclusions.

The success of the program depends on the quality of the analysis. A direct translation of the analysis of CATO [4] yields a success rate almost identical to that found for CATO in [10] (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 can suggest 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.

While the output from the program provided good diagnostics and a reasonable explanation of the result, it will require post-processing and some additional knowledge to put it into a form resembling 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 actual decisions.

Our approach has clear correspondences with structured argumentation [14]. These merit further exploration.

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. Also worthy of exploration are: producing a variant program to generate all the possible arguments and resolve conflicts explicitly, so as to facilitate comparison with ASPIC+; a variant to produce more lawyer-friendly explanations; and a variant to enable conceptual retrieval of cases.

References

[1] L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. Abstract dialectical frameworks for legal reasoning. In *Proceedings of Jurix 2014*, pages 61–70, 2014.

[2] L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. *Evaluating an Approach to Reasoning with Cases Using Abstract Dialectical Frameworks*. Technical Report ULCS-15-002, University of Liverpool, 2015.

[3] L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. Factors, issues and values: Revisiting reasoning with cases. In *Proceedings of the 15th ICAIL*. This volume, 2015.

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

[5] K. Ashley. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge, MA, 1990.

[6] K. Atkinson, T. Bench-Capon, H. Prakken, and A. Wyner. Argumentation schemes for reasoning about factors with dimensions. *Proceedings of JURIX 2013*, pages 39–48, 2013.

[7] T. Bench-Capon. Representing popov v hayashi with dimensions and factors. *Artif. Intell. Law*, 20(1):15–35, 2012.

[8] L. K. Branting. Reasoning with portions of precedents. In *Proceedings of the 3rd ICAIL*, pages 145–154. ACM, 1991.

[9] G. Brewka, H. Strass, S. Ellmauthaler, J. Wallner, and S. Woltran. Abstract dialectical frameworks revisited. In *Proceedings of 23rd IJCAI*, 2013, issued on CD.

[10] S. Brüninghaus and K. Ashley. Predicting outcomes of case-based legal arguments. In *Proceedings of the 9th ICAIL*, pages 233–242, 2003.

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

[12] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and *n*-person games. *Artif. Intell.*, 77:321–357, 1995.

[13] J. Horty and T. Bench-Capon. A factor-based definition of precedential constraint. *Artif. Intell. Law*, 20(2):181–214, 2012.

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

[15] H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game. *Artif. Intell. Law*, 6(2-4):231–287, 1998.

[16] D. B. Skalak and E. L. Rissland. Arguments and cases: An inevitable intertwining. *Artif. Intell. Law*, 1(1):3–44, 1992.