Abstract Dialectical Frameworks for Legal Reasoning

Latifa AL-ABDULKARIM, Katie ATKINSON, Trevor BENCH-CAPON
Department of Computer Science, The University of Liverpool, UK

Abstract.
In recent years a powerful generalisation of Dung's abstract argumentation frameworks, Abstract Dialectical Frameworks (ADF), has been developed. ADFs generalise the abstract argumentation frameworks introduced by Dung by replacing Dung's single acceptance condition (that all attackers be defeated) with acceptance conditions local to each particular node. Such local acceptance conditions allow structured argumentation to be straightforwardly incorporated. Related to ADFs are prioritised ADFs, which allow for reasons pro and con a node. In this paper we show how these structures provide an excellent framework for representing a leading approach to reasoning with legal cases.

Keywords. case based reasoning, factors, argumentation frameworks

1. Introduction

Modelling reasoning using legal cases has been a central concern since the beginning of AI and Law, and is also an important example of the study of computational argument applied to real problems. The leading approach to reasoning with legal cases in AI and Law is represented by HYPO [4], CATO [2] and IBP [13]. Cases are represented as sets of *factors*, legally significant patterns of facts which favour one side or the other. But there have also been attempts to formalise the reasoning, including use of Dung's abstract frameworks [15] in e.g. [18]. A significant development joining the empirical and formal approaches was [19], which represented precedent cases as three rules, one offering the pro-plaintiff factors as a reason to decide for the plaintiff, one giving the pro-defendant factors as a reason to decide for the defendant, and one stating the priority between these two rules, as determined by the actual decision in the case. Note here the rules include all available factors and so offer the strongest available reasons for plaintiff and defendant: a recent formalisation of this approach [16] allows for weaker rules for the winner. Such broader rules, if sufficient to outweigh the opposing reasons, permit *a fortiori* reasoning.

In recent years a powerful generalisation of Dung's abstract argumentation frameworks, Abstract Dialectical Frameworks (ADF), has been developed in [12] and [11]. ADFs generalise the abstract argumentation frameworks introduced by

Dung [15] by replacing Dung's single acceptance condition (that all attackers be defeated) with acceptance conditions local to each particular node: for example some nodes could be acceptable if at least one attacker were defeated. Such local acceptance conditions allow structured argumentation to be straightforwardly represented. Related to ADFs are prioritised ADFs (PADFs), which allow for both pro and con reasons. In this paper we will explore the use of this new formal framework for representing legal case based reasoning. As well as being useful for reasoning with cases, this allows the strengths of ADFs to be seen in the context of a significant application.

We define ADFs in section 2 and describe cased-based reasoning with factors in section 3. Section 4 relates the two and section 5 describes the advantages of doing so, and the next steps in our programme of work.

2. Abstract Dialectical Frameworks

Abstract Dialectical Frameworks (ADFs) were introduced in [12] and revisited in [11]. ADFs provide a generalisation of Dung's abstract argumentation frameworks (AFs) [15]. ADFs, like AFs, consist of a set of nodes and directed links between them, but whereas the links in an AF have a uniform interpretation, namely *defeat*, the links in an ADF can be given a variety of interpretations. Moreover in ADFs the nodes are *statements* in general, rather than specifically *abstract arguments*. ADFs are defined in ([11]) as follows:

> **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.

With respect to C it is stated in [11] that "In many cases it is convenient to represent acceptance conditions as propositional formulas." Later [11] defines a Prioritized ADF (PADF), in which the links are partitioned into support links (L^+) and attack links (L^-), and C is replaced by $>$, a strict partial order (irreflexive, transitive, antisymmetric) on S representing preferences among the nodes. This structure is intended to reproduce exactly the Preference Based Argumentation Frameworks of Amgoud and Cayrol [3], which augments an AF with an ordering on arguments. In [3] the preferences act like an oracle, and no explanation or rationale is offered for the preferences. Moreover, they are given in advance, not at run-time, so that the preference order cannot be argued for. In contrast, reasoning transparently with these preferences is the whole point of legal case based reasoning systems such as CATO and IBP. Attempts to justify preferences in abstract argumentation systems have been made. For example, in Value Based Argumentation [7] arguments are justified in terms of the values promoted by the arguments concerned. A general framework for arguing about preferences is provided in [17]. To represent legal CBR as modelled in AI and Law systems such as [4], [2] and [13] we need to go beyond [3] and provide acceptance conditions expressed in terms of precedent cases to enable the preferences so expressed to be justified by reference to precedent cases. Moreover, we will not wish to insist

that the ordering on S is global to the PADF, but allow it to be local, so that the influence of a factor may be properly contextualised. For example a particular factor may be relevant to several different issues, and so be linked to several distinct elements of S, but the strength and nature of its support may vary for these different nodes.

3. Factor Based Reasoning

The consensus that has developed in AI and Law is that legal reasoning passes through a series of stages, moving from evidence to a final decision. Cases begin with evidence. Evidence can, of course, conflict, and so it is necessary to resolve these conflicts to come to a set of agreed facts. This stage is often adjudicated by a jury rather than trained legal personnel, and the style of argumentation is rather different. Moreover facts are usually considered settled when the case comes before a higher court. For this reason, this stage is often considered as a separate topic (e.g. [9]). The facts of cases exhibit enormous variety, and if we are to make comparisons between them, so that the current case can be related to precedent cases, we need to abstract from the particular facts to some intermediate concepts. This idea goes back at least to Ross [20], but is also a key idea in contemporary AI and Law (see e.g. [5] for a discussion). These intermediate concepts are often termed *factors* and are stereotypical patterns of facts that are sufficiently abstract to apply to a good number of cases, and which have some significance in the body of case law, by favouring a particular side. In this stage the factors present in a case are identified, and this may itself be a subject of argument [6]. The most studied stage is the transition from factors to a legal decision. In [2], [8], [5] and [16], cases are represented as collections of factors, some favouring the plaintiff and some the defendant, which are then weighed to produce a decision.

In [2] factors are organised into a factor hierarchy. Base level factors, which are the factors used to describe a case, are grouped under more abstract factors. The presence of a factor in a case description is a reason why its abstract parent is or is not present in that case, and so these reasons need to be balanced when ascribing an abstract factor to the case. Several levels of abstract factor may be found in CATO. In IBP the organisation is taken further. The highest level of the hierarchy comprises not abstract factors but *issues*, and these form what is termed in [13] a *logical model*. The difference in [13] is that issues relate to the outcomes as a propositional function, rather than as a set of pro and con factors that need to be weighed. The logical model (which like CATO relates to US Trade Secret law) of [13] is shown in Figure 1.

This model can then be expanded downwards using the factor hierarchy of CATO. Nodes can be merged since the leaves of the IBP logical model can be made to correspond to abstract factors within the CATO hierarchy. Consider the extract from CATO's factor hierarchy shown in Figure 2. *Confidential Relationship* (F114) maps directly, as does *Improper Means* (F110). So does F101 (*Info Trade Secret*), if we identify *Information-Unique* with F104 (*Info Valuable*), which seems plausible given that *Unique Product* is its only factor. *Info-Used* does not appear in the extract shown in Figure 2, but is included as an abstract

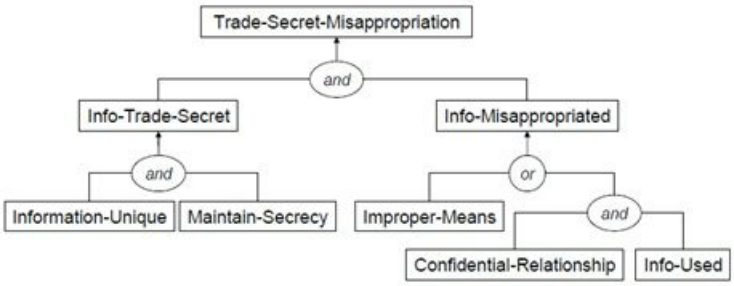

Figure 1. IBP Logical Mode from [5]

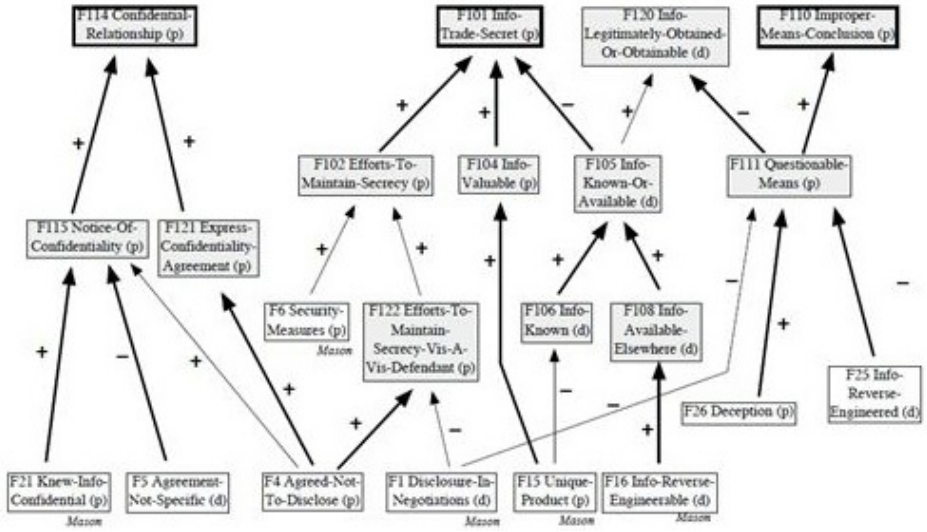

Figure 2. Extract From CATO Abstract Factor Hierarchy from [2]

factor (F112) in the complete hierarchy given in [2]. Although the full factor hierarchy also contains additional base level factors, they do not raise any different questions, and so we need not consider them here.

The hierarchy can be extended further downwards, so that we can make the facts on which the factors depend explicit. The need to be able to argue about which factors should be used to represent cases was the topic of [6]. Information about the facts which relate to factors can either be gleaned from the factor descriptions in Appendix 2 of [2], or by going back to the original case reports (cf. [1]), and will be similar to that contained in the *focal slots* of HYPO [4].

Thus the hierarchy begins with issues: these form the logical model. Issues are either *satisfied* or *unsatisfied* in particular cases, and the links between them taken from the logical model are all "+". At their lowest level, issues have links to factors. Factors are either present or absent in a case, and upwards links to issues (and more abstract factors) can be either "+" or "-", so that we must regard the

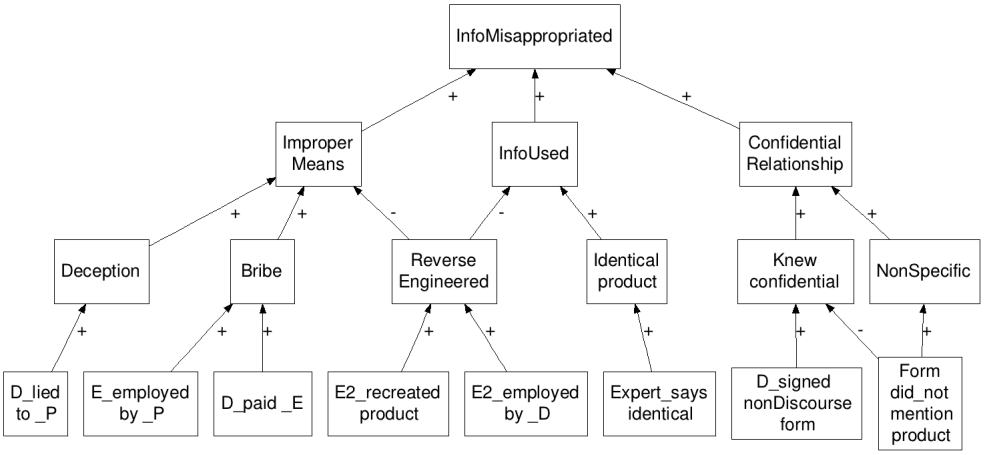

Figure 3. Prioritied Abstract Dialectical Framework

structure as a PADF. The downwards links of the lowest level (base level) factors are to facts and may also be either "+" or "-". Facts are *true* or *false* with respect to a given case, and form the leaves of the structure.

As an example of a PADF for reasoning about US Trade Secrets, consider Figure 3. This is composed from extracts from the logical model of IBP, the factor hierarchy of CATO, and some possible facts to justify the presence of factors. The issue is *Info Misappropriated*. Factors are *Improper Means*, *Info Used and Confidential Relationship*, *Deception*, *Bribe*, *Reverse Engineered*, *Identical Product*, *Knew Confidential* and *Non Specific*. The remaining nodes are facts.

4. Relating to Prioritied Abstract Dialectical Frameworks

We can now explain the relationship between the diagram in Figure 3 and the PADF as defined in [11]. The issues, factors and facts form the set S. Links are those in the figure, partitioned into "+" and "-" links as required by the definition. All that remains is to define the acceptance conditions. We will specify the acceptance conditions using Prolog procedures, which provide a set of clauses each giving a sufficient condition for or against acceptance, expressing priority through the ordering of these clauses. Propositional functions are represented using Prolog's declarative semantics and priorities expressed using Prolog's procedural semantics. Priorities are needed since in legal CBR it is not always possible to supply necessary and sufficient conditions. The clauses are formed from the children of a node: the "+" links are reasons why an issue (factor, fact) is satisfied (present, true) and the "-" links are reasons why an issue (factor, fact) is unsatisfied (absent, false). Taken together the links of a given polarity form the body of a clause, as does every subset of the links of a given polarity. Finally the empty set is also a clause giving the default value, which, in the context of reasoning with cases, indicates the burden of proof. Thus in the case of *Info Misappropriated* (abbreviated to *infoMis* in the code to fit the page width) where we have three positive links in Figure 3, we get:

```
issue(infoMis,satisfied):- issue(improperMeans,present).
issue(infoMis,satisfied):- issue(confidentialRelationship,present).
issue(infoMis,satisfied):- issue(infoUsed,present).
issue(infoMis,satisfied):- issue(infoUsed,present),
                           issue(confidentialRelationship,present).
issue(infoMis,satisfied):- issue(improperMeans,present),
                           issue(confidentialRelationship,present).
issue(infoMis,satisfied):- issue(improperMeans,present),
                           issue(infoUsed,present).
issue(infoMis,satisfied):- issue(improperMeans,present),
                           issue(infoUsed,present),
                           issue(confidentialRelationship,present).
issue(infoMis,unsatisfied).
```

We now use the logical relations of the logical model to identify which of these clauses are required. The two disjucts in the logical model correspond to the first and fourth clauses, and the last clause provides the default. We can disregard the rest since they are either subsumed or insufficient. Thus the acceptance condition for *Info Misappropriated* is:

```
issue(infoMis,satisfied):- issue(improperMeans,present).
issue(infoMis,satisfied):- issue(infoUsed,present),
                           issue(confidentialRelationship,present).
issue(infoMis,unsatisfied).
```

Turning to factors we typically have reasons for both presence and absence of a factor, but until we consider the precedent cases we do not know which reasons should prevail. Thus at the outset we have, for *InfoUsed*:

```
factor(infoUsed,unknown):-factor(identicalProduct,present),
                          factor(reverseEngineered,present).
factor(infoUsed,present):-factor(identicalProduct,present).
factor(infoUsed,absent):-factor(reverseEngineered,present).
factor(infoUsed,absent).
```

This means that we can say whether *InfoUsed* is present or absent if we have one, or neither, of the children factors, but if both are present the status of *InfoUsed* is unknown. This is why we need the precedent cases. Suppose there is a precedent with both factors present in which it was decided that the information was not used. We might then conclude that *reverseEngineered* outweighs *identicalProduct* and express this as a priority between the second and third clauses:

```
factor(infoUsed,absent):-factor(reverseEngineered,present).
factor(infoUsed,present):-factor(identicalProduct,present).
factor(infoUsed,absent).
```

Note, however, that we cannot remove the first clause with certainty unless we consider the case as a whole: if both of these factors are present and the case was found for the plaintiff, this could be because the *ImproperMeans* factor was

also present. Since this issue is itself enough to justify finding for the plaintiff, the information was misappropriated whatever the status of *InfoUsed*, and so this precedent cannot establish priorities between factors related to that issue. Thus our precedent should also be such that *improperMeans* is absent and *confidentialRelationship* is present.

It may be that our precedents suffice only to provide partial acceptance conditions. Consider *ImproperMeans*, and suppose we have a precedent with both *bribe* and *reverseEngineered*, allowing us to see that bribe has priority, but no precedents with both *deception* and *reverseEngineered*. Our acceptance condition will now be:

```
factor(improperMeans,present):-factor(bribe, present).
factor(improperMeans,unknown):-factor(deception, present),
                                factor(reverseEngineered, present).
factor(improperMeans,absent):-factor(reverseEngineered, present).
factor(improperMeans,present):-factor(deception, present).
factor(improperMeans,absent).
```

The ordering of the third and fourth clauses does not matter since, if both are present, the second clause will apply and the third and fourth will not be tried. Once we have a suitable precedent with both *deception* and *reverseEngineered* present we can order the third and fourth clauses in accordance with the decision in that case and discard the second clause.

The base level factors are resolved similarly, but with facts in the bodies of the clauses. Very often these will be sufficient conditions. In Figure 3 apart from *knewConfidential*, all the facts offer a single sufficient condition, comprising one or two facts. We could easily add others: for example a defendant may deceive by giving the plaintiff to understand that he represents a particular company without an explicit lie, or there could be other ways of showing that two products were identical. The acceptance condition for *knewConfidential* is a bit more interesting. The intention here is that if the defendant has signed a non disclosure form it shows that he knew the information to be confidential, unless there is no specific mention of the information in the form, in which case he may argue that he did not know this particular information was confidential. This defence presupposes that there was a form which did not mention the information: thus we need:

```
factor(knewConfidential,absent):- fact(signed,form,defendant),
                                  fact(notMentioned,form,product).
factor(knewConfidential,present):- fact(signed,form,defendant).
factor(knewConfidential,absent).
```

Essentially we have here a form of exception. The reasons are not independent. The question of whether the product was specifically mentioned arises only if there was a signed form, and provides a possible exception to the general rule expressed in the second clause. Note that in Prolog the *most* specific clause appears first.

5. Discussion and Future Work

5.1. Advantages

There are three types of advantage of considering case-based reasoning in terms of PADFs. There are advantages in terms of the theoretical aspects of computational argumentation, advantages in terms of the expressiveness of the representation, and practical advantages in terms of building systems.

The theoretical advantages arise once we are able to see the structures required for legal case-based reasoning as ADFs because we are able to take advantage of the results developed for ADFs and apply them to legal case-based reasoning. Thus for example, the correspondences between ADFs and AFs identified in [11] can be directly carried over to factor-based reasoning. Equally the complexity results for ADFs in [11] can be applied to legal CBR systems. There is much current activity in the theoretical argumentation community relating to ADFs and seeing legal CBR in terms of ADFs allows the fruits of these endeavours to be enjoyed by the AI and Law community also.

The advantages with respect to expressiveness come from the more natural representation offered by ADFs especially where we have both pro and con reasons. Using AFs we are restricted to only one kind of acceptance condition: the parent is accepted if, and only if, all the children are rejected: that is, the children represent a disjunction of potential defeaters. In order to represent the acceptance condition for InfoMisappropriated using standard AFs, we would need to rename ImproperMeans *properMeans*, and introduce a node for *noConfidentialInformationUsed* attacking InfoMisapprpriated and attacked by InfoUsed and ConfidentialRelationship. The factors thus have to be expressed in an unnatural fashion to give them the right polarity, and additional factors introduced to artificially link conjuncts. Moreover, whereas the support between elements needs to be represented as an attack on an attacker in a standard AF (so introducing an additional element and disguising the supportive nature of the relationship), support is expressed directly in a PADF by using a "+" link. This is not a criticism of AFs for their original purpose, but rather a consequence of AF nodes being *arguments*, whereas ADFs nodes are statements, with the arguments encapsulated in the acceptance conditions. Since we are dealing with the satisfaction of issues, the presence and absence of factors and the truth of facts, we think that ADFs, or more particularly PADFs, are better suited for our purposes. The key to this is the flexibility of PADFs, which means that the most straightforward representation can be used, without the need to contort the information into a prescribed form. Additionally, PADFs provide an excellent way of handling the need to reason with portions of precedents, identified by Branting [10], but never really satisfactorily addressed. This is what enables us to go beyond the *a fortiori* reasoning of [19] without resorting to the apparently arbitrary choices of [16]. Moreover, as we have seen above, PADFs can be used to represent uniformly not only propositional functions, but, where necessary and sufficient conditions cannot be specified, reasons for and against, which need be weighed to reach a decision, also.

The practical advantages step from the closeness of the representation to an executable form. Because the acceptance conditions are represented as Prolog

procedures, we can regard the set of acceptance conditions as a Prolog program. All nodes are associated with a procedure with that node as the head of the clauses and its children form the bodies. These children will have acceptance conditions comprising procedures in which they are the heads and their children form the bodies. Thus each level is defined in terms of the next level down. When we reach the leaves we are confronted by facts. These can be supplied using whatever interface we choose: a set of cases from a file; interactively using a form or by asking questions; or even, if suitable software is available, by extracting the facts from a description of the case. Thus the move from the analysis to an executable program is direct and immediate: debugging, modification and revision can be performed on the representation and then mechanically transferred to the executable program, with all the software engineering benefits that this affords.

5.2. Planned Evaluation

The initial evaluation will be only on the issue and factor part of the structure as this can draw on currently available analysis, and enables direct comparison with existing systems. Because the full 148 case data set used in [13] is not publicly available, we will use the the subset of 33 cases collected from published sources by Alison Chorley and used to evaluate the Agatha system in [14].

We will take the nodes and links for our PADF from the logical model of [13] extended through abstract factors to base level factors in Figures 3.2 and 3.3 of [2]. We will then perform a set of tests using the cases from [14]. The results can then be compared with those achieved by Agatha in [14] and by IBP in [13]. Moreover the evaluation of IBP also tested a version of CATO (NoSignDist-BUC which used the best untrumped case, allowing CATO's factor hierarchy to be used to downplay distinctions where appropriate) and a version of HYPO (Hypo-BUC, which used the best untrumped case with a flat set of factors). Both Agatha and IBP achieved a success rate of over 90%, with very few abstentions. NoSignDist-BUC made more errors and had more abstentions, giving a success rate of 77.8%. Hypo-BUC made fewer errors than either, but abstained in a very large number of cases, so that its success rate was only 66.3%. Because our program will benefit from the structure taken from IBP, and also recognises the need to consider the contribution of a factor in the context of particular issues, we would expect the performance to be comparable to that of IBP and Agatha.

The next step will be to supply the fact layer. We can take some guidance from the focal slots of HYPO, but in the main we will need to return to the original decisions and oral transcripts which we will analyse using the methods of [1]. Our expectation is that this will achieve better results than the program which stops at factors, since taking the ascription of factors on trust may miss some important nuances.

5.3. Concluding Remarks

The Abstract Dialectical Frameworks of [11] provide a powerful generalisation of Dung's Abstract Argumentation Frameworks [15]. In this paper we have shown that they provide a natural way to express in a formal manner reasoning with

legal cases using factors developed over the years through important practical systems such as HYPO, CATO and IBP.

References

[1] L. Al-Abdulkarim, K. Atkinson, and T.Bench-Capon. From oral hearing to opinion in the us supreme court. In *Proceedings of JURIX 2013*, pages 1–10, 2013.

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

[3] L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 1–7, 1998.

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

[5] K. Ashley and S. Brüninghaus. A predictive role for intermediate legal concepts. In *Proceedings of JURIX 2003*, pages 1–10, 2003.

[6] K. Atkinson, T. Bench-Capon, H. Prakken, and A. Wyner. Argumentation schemes for reasoning about factors with dimensions. *Proceedings of 26th International Conference on Legal Knowledge and Information Systems (JURIX 2013)*, 2013.

[7] T. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[8] 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.

[9] Floris J Bex. *Arguments, stories and criminal evidence: A formal hybrid theory*, volume 92. Springer, 2011.

[10] K. Branting. Reasoning with portions of precedents. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, pages 145–154, 1991.

[11] G. Brewka, H. Strass, S. Ellmauthaler, J.P. Wallner, and S. Woltran. Abstract dialectical frameworks revisited. In *23rd International Joint Conference on Artificial Intelligence*, 2013.

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

[13] S. Brüninghaus and K. Ashley. Predicting outcomes of case-based legal arguments. In *9th International Conference on Artificial Intelligence and Law*, pages 233–242, 2003.

[14] A. Chorley and T. Bench-Capon. An empirical investigation of reasoning with legal cases through theory construction and application. *Artif. Intell. Law*, 13(3-4):323–371, 2005.

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

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

[17] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173(9-10):901–934, 2009.

[18] H Prakken. From logic to dialectics in legal argument. In *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, pages 165–174, New York, NY, USA, 1995. ACM.

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

[20] A. Ross. Tu-tu. *Harvard Law Review*, 70:812–825, 1961.