Implementing a Theory of a Legal Domain

Trevor BENCH-CAPON [a] and Thomas F GORDON [b]
[a] *University of Liverpool, UK*
[b] *Berlin, Germany*

Abstract. We describe a system for constructing, evaluating and visualising arguments based on a theory of a legal domain, developed using the Angelic methodology and the Carneades argumentation system. The visualisation can be used to explain particular cases and to refine and maintain the theory. A full implementation of the well known US Trade Secrets Domain is used to illustrate the process.

Keywords. legal argumentation, Angelic methodology, Carneades

1. Introduction

Modelling reasoning with legal cases has been a central concern of AI and Law from the very beginning. A good deal of the research has built on the pioneering work of HYPO [1] and CATO [2]. Both of these addressed cases in US Trade Secrets law, which is the domain which we will model in this paper. Over the years a series of stages in the reasoning have been identified. The outcome is decided on the basis of the resolution of a number of issues which set out what must be shown to establish a claim. The relationship between the issues and the outcome can be expressed in a set of rules [3]. The issues are resolved by weighing the reasons to resolve that issue for the plaintiff against the reasons to resolve that issue for the defendant (e.g. [4]). These reasons are generally called *factors* [2], and the preferences between them are (in common law domains) derived from the decisions in previous, precedent, cases [5]. Often the set of factors describing the cases is taken as given, but in some cases the ascription of factors is itself a matter of controversy which must be resolved using precedents [6]. This structure lends itself to a hierarchical representation, as in [2], with the outcome as the root, issues at the upper layers, abstract factors in the middle layers and the base level factors as the leaves. This was extended in [7] to allow the base level factors to be ascribed on the basis of a series of questions answered by the user.

We will describe how to realise this approach to produce a system which will model the domain theory and, when given a particular case, will produce an argument map showing what was accepted and what was rejected in the case, and the arguments which justified these positions. The argument map provides a visual explanation of the reasoning in the case, and where the result is unexpected provides a means to identify how to perform the required corrective or adaptive maintenance of the theory.

Section 2 describes the methodology used to specify the domain theory and the system used to make the theory executable. The transition from theory to implementation is

Table 1. Sample fragment of ADP for Trade Secret Misappropriation from [9]

Node	Children	Acceptance Conditions	Justification
InfoValuable	F6p F8p F11d F15p InfoObtainable	REJECT IF F11d ACCEPT IF F8p ACCEPT IF F6p ACCEPT IF F15p REJECT IF InfoObtainable ACCEPT	Silfen Lewis Mason College Restatement
InfoObtainable	F15p F16d F20d F24d	REJECT IF F15p ACCEPT IF F16d OR F24d OR F20d REJECT	College Ferranti

described in Section 3, while Section 4 discusses how the argument map can be used for explanation and maintenance. Finally, Section 5 offers some discussion and concluding remarks.

2. Background

2.1. The Angelic Methodology

The Angelic methodology, which is intended to produce a theory of a legal domain, was introduced in [8] and has subsequently been refined in a series of projects with the law firm Weightmans, including one described in [7]. The methodology has two outputs: the Angelic Design Proforma (ADP), originally called ADF, and a set of questions required to instantiate the ADP for a particular case.

The ADP comprises a table with four columns, in which the rows describe nodes in a hierarchy. Example rows, taken from the full example given in [9], are given in Table 1. The first column gives the ID of the node, which is intended to be an informative label for the node. The second column gives the children of the node. The third column gives a list of prioritised reasons to accept or reject the node. These reasons use only the children of the node, and conclude with a default in case none of the reasons are satisfied. The final column gives the source of these reasons and priorities between them which may be a statute, a commentary, a precedent case or any other authoritative source.

The hierarchy is essentially the factor hierarchy of CATO [2], with the outcome as the root, issues at the upper layers, abstract factors in the middle layers, and base level factors as the leaves. The acceptance conditions are inspired by Abstract Dialectical Frameworks [10]. The questions are intended to be posed to the user to give the leaf nodes for a particular case. A leaf node may correspond to a single answer, or be derived from one or more answers. An example question is discussed in Section 3.2.

2.2. Carneades

Carneades is both a formal model of structured argument and a software implementation of this formal model. The original version of Carneades [11] provided a recursive procedure to evaluate argument graphs, given a set of assumptions, to label the statement nodes acceptable (In) or not acceptable (Out), and a way to visualise the output in "argument maps" [12].

Here We are using the latest version of Carneades (4.3). The formal model of this version was first presented in [13]. While it supports all the features of the earlier version, the formal model is quite different. These differences were motivated by the desire to handle not just attack relations among arguments, but also the *balancing* of competing arguments, so as to better be able to support practical reasoning, where the pros and cons of alternative options are weighed, including support for multi-criteria decision analysis (MCDA). This also enables support for cumulative arguments, where the failure of a premise can weaken an argument without defeating it completely. To achieve these goals, the structure of argument graphs is now tripartite, with issue, statement and argument nodes, and arguments have been extended with assignable and customisable weighing functions [14]. When evaluating argument graphs, statements are now labelled either In, Out or Undecided. The formalisation of issues assures that at most one option (position) is In, serving as a constraint. The formal model now uses fixed-point semantics when evaluating argument graphs, so as to be able to handle cycles in argument graphs. Just as in abstract argumentation, various semantics can be applied, such as grounded, preferred or complete semantics. The implementation, however, only supports grounded semantics, which we have found to be sufficient for our legal application scenarios. As before, the implementation provides a way to visualise argument graphs in argument maps.

Another new feature of this version of Carneades is its provision of a language and inference engine for argumentation schemes. The language is based on Constraint Handling Rules [15], a forwards-chaining rule system with a declarative, logical semantics. In our implementation of Constraint Handling Rules [16], every time a rule is applied (fired), an argument is generated as a side-effect. Given a set of argumentation schemes (rules) and a set of assumptions, the rule engine is first applied to generate an argument graph and then the argument graph is evaluated to label the statements in the graph. Finally, as before, the resulting argument graph can be visualised in an argument map.

3. Realising an Angelic Theory in Carneades

To make the theory executable it is necessary to represent the acceptance conditions and to get the assumptions from the user to instantiate a particular case. Because the underlying conceptions of Angelic and Carneades are very similar, both conceptualising the reasoning as forming an argument graph, each acceptance condition can be represented as a Carneades scheme. For moving from question responses to factors we distinguish three kinds of factors, as discussed in Section 3.2.

3.1. Representing the acceptance conditions

Let us use the acceptance conditions for the InfoObtainable node of the trade secrets ADP, shown in Table 2 below, as an example. Each node in the ADP has four properties (columns in the table): a node name; the children of the node, used in the acceptance conditions; a set of prioritised acceptance conditions; and the source of each condition, the case or text on which it is based. As we will see, all of this information in an ADP maps directly into Carneades argumentation schemes. The first acceptance condition can be represented as follows:

```
id: ioCollege
meta:
    source: College
weight:
    constant: 1.0
conclusions: [notio]
premises: [f15p]
```

Here, `ioCollege` is an identifier for the scheme. The `meta` property can be used to annotate the scheme with any desired information. Here we have used it to provide the source of the scheme, the precedent case *College Watercolor Group, Inc. v. William H. Newbauer, Inc* (*College*). The `weight` property is used only to order the schemes for a node, so that its particular value has no other significance. Here we are using constant weights for this purpose. Carneades also provides a variety of weighing functions which can be used to compute weights. This feature will be demonstrated later, when showing how factors with magnitude and dimensions can be handled. Using weights, arguments can be partially ordered, useful for giving acceptance conditions with alternative premises the same weight, as will be demonstrated below. The conclusion of this scheme, `notio`, means that the information was not obtainable, and denotes the negation of `io`, that the information was obtainable. The conclusion indicates whether the node in the ADP being represented (InfoObtainable) is accepted or rejected. The premise of this example scheme is `f15p`, denoting the F15p factor, Unique-Product, the body of the acceptance condition. Schemes may have multiple conclusions and premises, although this feature is not demonstrated in this example.

To make `io` and `notio` conflict, so that at most one of them can be labelled In, an issue scheme is added as follows:

```
issue_schemes:
    io: [io, notio]
```

To complete this example, the schemes for the remaining acceptance conditions for the InfoObtainable node of the ADP in Table 2 are shown in Figure 1:

The constant weights used in these schemes simply enforce the ordering of the acceptance conditions in the ADP. Any real numbers could have been used, so long as they preserve the desired ordering. Notice that two schemes were needed to represent the acceptance conditions for the *Ferranti* precedent, since it has two alternative premises, F24d OR F20d. Both of these schemes were given the same weight, 0.9, since they share the same position in the ordered list of acceptability conditions in the ADP.

```
- id: ioFerranti1
  meta:
      source: Ferranti
  weight:
      constant: 0.9
  conclusions: [io]
  premises: [f24d]
```

```
- id: ioFerranti2
  meta:
     source: Ferranti
  weight:
     constant: 0.9
  conclusions: [io]
  premises: [f20d]
```

```
- id: ioDefault
  weight:
     constant: 0.1
  conclusions: [notio]
```

Figure 1. Argumentation Schemes for Information Obtainable

3.2. Moving from facts to factors

Most factors are simply Boolean and depend on a single fact, and so they can be assumed directly on the basis of particular question answers. Other factors require some judgement to be ascribed to cases on the basis of the facts. These are the factors with magnitude [17]. Two types of such factors are used: those derived from a single dimension and which apply a threshold to determine whether the factor applies and those (e.g. CompetitiveAdvantage in CATO) ascribed on the basis of a weighted sum of two dimensions.

We illustrate this using the question relating to disclosures.

> Q3 Was the Information disclosed (Check all that apply)
> (a) In negotiations with the Defendant?
> (b) To employees?
> (c) To sub-contractors?
> (d) To customers?
> (e) To the public?
> (f) Restrictions were placed on the disclosures
> (g) The information was not disclosed

Answers (a) and (f) lead directly to factors F1d and F12p respectively, while (g) leads to no factor. Answers (b)-(e) represent the four points on the dimension leading to the ascription of F10p (InfoNotDisclosed) and F10d (InfoDisclosedToOutsiders). Which factor applies depends on where the threshold is drawn, in the light of precedents and other domain knowledge.

3.2.1. Thresholds

In the trade secrets domain, answers (b)-(e) to the question about whether information has been disclosed to outsiders form a *dimension*, which can be satisfied to a greater or lesser extent. Let us identify the following points along the disclosure dimension, based on to whom the information has been disclosed: employees, subcontractors, customers and the public. These values will need to be mapped to factors, to show which party is favoured on the dimension. Where the line is drawn is established in precedents. Based on an analysis of the cases, we want to map disclosures to employees and subcontractors to the factor F10p (InfoNotDisclosed) but map disclosure to customers and the public to F10d (InfoDisclosedToOutsiders), where F10p and F10d are alternative positions of an issue, so that at most one of these factors may be In. Effectively, the dimension will be partitioned with a threshold between the subcontractors and customers points. Moreover, we want to use argument weights to preserve the information about the relative position of the disclosure along the dimension. So that, for example, a disclosure to the public is a stronger argument for F10d than a disclosure to customers. And, conversely, so that a disclosure to employees is a stronger argument for F10p than a disclosure to subcontractors. One way to achieve these goals, using the multi-criteria decisions analysis (MCDA) weighing function provided by Carneades is shown in Figure 2.

This use of the MCDA weighing function does not illustrate its full capabilities, since there is only one dimension, “disclosed”, where typically MCDA will be used to combine multiple dimensions using a weighted sum. Notice that the permitted values for the disclosed dimension (property) differ between these two schemes. The customers and public values are in the first scheme, and the employees and subcontractors values are in the second scheme. Alternatively, one could list all the values along the disclosure

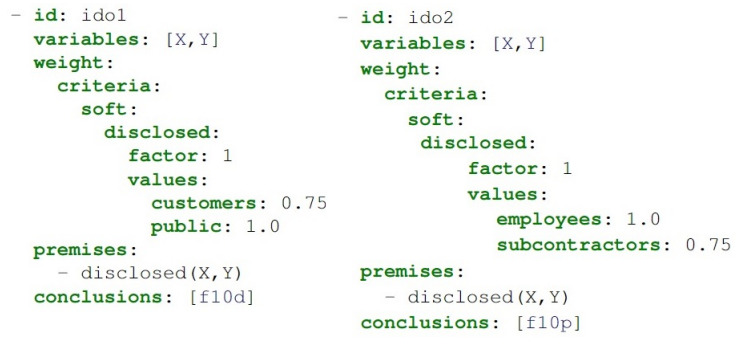

Figure 2. Argumentation Schemes for Discloses To Outsiders

dimension in both schemes, but assign the weight 0.0 to the other values. Notice also that the argument weights do not increase from the beginning to the end of the dimension, but rather increase from the threshold in the middle out to the points on either end of the dimension. Finally, notice that the disclosed property declares a binary predicate, `disclosed/2`. Although not needed for this application, this feature is used when addressing practical reasoning. In the next section, we make fuller use of the expressivity of the MCDA weighing function to represent the mapping of two dimensions to a factor.

3.2.2. Weighted Sums

In the trade secrets domain, whether or not the defendant has obtained a competitive advantage due to acquiring knowledge of the trade secret is a function of two dimensions, the costs saved and the time saved by the defendant. Fewer time savings can be compensated by greater cost savings, and vice versa. If both time and cost savings have been substantial, the argument for having obtained a competitive advantage will be greatest.

Here we demonstrate how to map several dimensions (time and cost savings) to an abstract factor, (F9d, NoCompetitiveAdvantage) using a Carneades scheme with an MCDA weighing function shown in Figure 3.

The weight of the argument is computed by a weighted sum of the values for each dimension. The relative weight of each dimension is given with a "factor". In this example, time saved is given twice the weight of costs saved. Any integer or real number may be assigned as a weight. The weighted sum is determined by first computing the relative portion of each factor compared to the sum of the factors for all the dimensions. Thus, in this example, time gets 2/3 of the weight and costs 1/3.

The second scheme here, with the id `caThreshold`, sets the threshold weight which an argument for F9d (NoCompetitiveAdvantage) must have in order to succeed. We have set the threshold at 0.5, meaning that the weighted sum of the time and cost savings properties must be greater than 0.5 in order for F9d to be derived (labelled In). Otherwise F8p (CompetitiveAdvantage) will be In.

4. Explanation From The Argument Map

The graph can be used to construct an explanation according to the well known Issue-Rule-Application-Conclusion method, widely used in US Law Schools and advocated

```
- id: ca1
  variables: [X,P,T]
  weight:
    criteria:
      soft:
        costs_saved:
          factor: 1
          values:
            very_large: 0.0
            significant: 0.25
            small: 0.5
            more_expensive: 0.75
        time_saved:
          factor: 2
          values:
            very_large: 0.0
            significant: 0.25
            small: 0.5
            took_longer: 0.75
  premises:
    - costs_saved(X,P)
    - time_saved(X,T)
  conclusions: [f9d]
```

```
- id: caThreshold
  weight:
    constant: 0.5
  conclusions: [f8p]
```

Figure 3. Argumentation Schemes for Competitive Advantage

in [18]. In this context the issue is the main point of contention in the case, rather than the top level nodes of the CATO hierarchy: thus any node may be the issue. A red node with a green child will indicate an issue, because the alternative position will have been preferred. In the case of *MBL (USA) Corp. v. Diekman* the relevant node is InfoValuable (see Figure 4). This is supported by InfoKnownToCompetitors, but excluded because SecurityMeasures is preferred, citing *Mason v. Jack Daniel Distillery* as a precedent. The issue is thus *whether the information is valuable if known to competitors when security measures were taken*, and the rule from *Mason*, as modelled in [9], is that it is, so we can conclude that the information is valuable in *MBL*.

Sometimes, an issue will appear at the level of factor ascription. Consider *Arco Industries Corp. v. Chemcast Corp*: all the base level factors favour the defendant and establish that the information was not a trade secret. We must therefore look for a factor with magnitude, in which the facts give a relatively low weight to the ascription. Here we find that DistinctProducts was assigned even though there was some resemblance. The issue here is *whether the products should be considered distinct where there is some resemblance*. In the decision, this was indeed the main point at issue, and it was decided that, despite some similarities, the defendant's product did not have an "indentation below the planar surface of the grommet which in turn lies below the peripheral sealing ridge", and this was enough to consider them distinct. A second issue raised in the case concerned disclosure to outsiders, and here it was found that the plaintiff had indeed disclosed to outsiders, even though the outsiders concerned were restricted to customers. Thus the issue *does disclosure to customers count as disclosure to outsiders* was also resolved in favour of the defendant. Once these factors have been described, there were no more points for the plaintiff to argue.

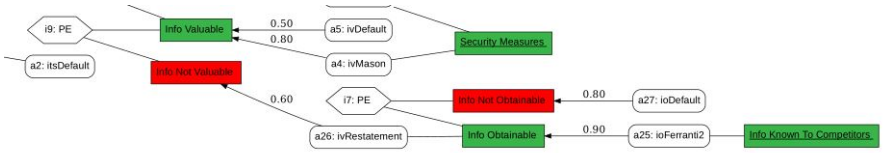

Figure 4. Argument map for the issue in MBL as modelled in [9], shown in Table 1

.

4.1. Refinement and Maintenance

It is rare that the ADP will be right first time, and so the development process will include running a set of test cases, and refining the ADP in the light of wrongly decided cases until all cases are explained correctly, or can be rejected as overruled. But even then the new decisions must be monitored, so that the ADP and its realisation can be maintained as the case law evolves. The visual representation in argument maps greatly supports the refinement and maintenance of the theory recorded in the ADP, by identifying the nodes which need attention.

We began this section with an explanation for the outcome of *MBL* based on the ADP of [9], where the information was considered valuable, even though known to competitors, because security measures had been taken. This was a perfectly good explanation, but in fact *MBL* was decided for the *defendant*, and so the explanatory preference is incorrect. This problem emerged when testing the system, but the procedure would be the same if maintaining the representation in the light of new decisions.

Having identified the issue on which the case turned, whether the misappropriated information was valuable, we can examine the relevant nodes in the ADP. In this case they are the nodes shown in Table 1. It can be seen from the InfoValuable node that F6p, SecurityMeasures, is preferred to InfoObtainable, in virtue of the precedent *Mason*. In *MBL*, InfoObtainable is accepted in virtue of F20d, InfoKnownToCompetitors, whereas in *Mason* it was accepted on the basis of F16d, InfoReverseEngineerable. As is clear from the second node in Table 1, F16d and F20d are given equal priority.

The solution is to recognise the lesser status of F16d by removing it from InfoObtainable: that the information may be discoverable by reverse engineering is clearly significantly weaker than it actually being known to competitors. Because it was the relative weakness of F16d that gave InfoObtainable its low priority in InfoValuable in Table 1, we can now increase the priority of InfoObtainable to reflect the decision in *MBL*. F16d remains with the original low priority, as derived from *Mason*. The revised nodes are shown in Table 2. Note that the modular design means that changes can be made to these nodes confident that other nodes will not be jeopardised.

In the case of *The Boeing Company v. Sierracin Corporation*, the defendant argued that the information had been disclosed to outsiders (see Figure 5). Here the issue is whether *if the information is disclosed to subcontractors, it is considered to have been disclosed to outsiders*. The current weights give the answer "no", but had *Boeing* been found for the defendant, this would need to be revised so that subcontractors were now included in outsiders when ascribing this factor.

Table 2. Fragment of ADP revised in the light of MBL

Node	Children	Acceptance Conditions	Justification
InfoValuable	F6p F8p F11d F15p F16d InfoObtainable	REJECT IF F11d ACCEPT IF F8p REJECT IF InfoObtainable ACCEPT IF F6p ACCEPT IF F15p REJECT IF F16d ACCEPT	Silfen Lewis MBL College Restatement Mason
InfoObtainable	F15p F20d F24d	REJECT IF F15p ACCEPT IF F24d OR F20d REJECT	College Ferranti

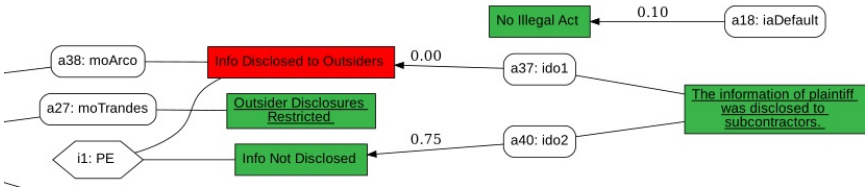

Figure 5. Fragment of Boeing showing disputed factor ascription

5. Discussion and Concluding Remarks

Angelic and Carneades are natural partners, since both conceptualise reasoning as moving from facts to conclusion through a series of arguments. Using Carneades to realise the acceptance conditions of an ADP encapsulating a theory of a legal domain is very straightforward: the node ID (or its negation) supplies the conclusion, the children in the body of the condition the premises, the weight indicates its priority and the source is also recorded. Factors with magnitude are also straightforwardly implemented through comparing weights as described in Section 3.2. This contrasts very favourably with previous implementations which required hand crafted code (e.g [19]). Moreover the argument schemes are held in a file separate from the program which executes them. This, given the simple correspondence between the schemes and the acceptance conditions, means that they can readily be edited to refine and maintain the theory by a knowledge engineer without any particular programming expertise.

A second major advantage of using Carneades to implement the ADP is that Caneades outputs an argument graph, directly supporting a visual representation rather than the textual explanations produced in previous work such as [8] and [20]. Visual presentations of argument have been popular since the use of Toulmin's argument scheme, and make the reasoning far easier to follow than working through a sometimes lengthy set of textual conclusions. The colour coding in Carneades makes it especially easy to identify the main issues, the key points of contention in the case, essential for deploying IRAC explanations, and to locate problems in the theory.

Although our focus in this paper has been on visualising cases, we would also like to point out that the system can be run in batch mode, so that a large number of test cases can be run quickly, generating argument maps for all of them at once. This enables any

failing cases to be identified for visual examination and correction. Equally this could be helpful for to comparing the performance of competing theories expressed as ADPs.

Our theory has been validated by correctly deciding all the test cases taken from the available literature. The method is not limited to the common law or theories derived from cases, but can be used to implement legal theories derived from any source texts.

Taken together, the Angelic methodology and the Carneades argumentation system provide a means of producing an executable theory of a legal domain drawn from leading cases, statutes, commentaries and experts, in a way which can readily be transformed into executable form to produce an evaluation and visual representation of the arguments in a particular case. As such it provides an excellent tool to support the analysis of a domain, refinement of the resulting theory and its maintenance as the law evolves.

References

[1] Rissland EL, Ashley KD. A case-based system for Trade Secrets law. In: Proceedings of the 1st ICAIL; 1987. p. 60-6.

[2] Aleven V. Using background knowledge in case-based legal reasoning: a computational model and an intelligent learning environment. Artificial Intelligence. 2003;150(1-2):183-237.

[3] Skalak DB, Rissland EL. Arguments and cases: An inevitable intertwining. AI and Law. 1992;1(1):3-44.

[4] Prakken H. A formal analysis of some factor-and precedent-based accounts of precedential constraint. Artificial Intelligence and Law. 2021;29(4):559-85.

[5] Horty JF, Bench-Capon T. A factor-based definition of precedential constraint. AI and Law. 2012;20(2):181-214.

[6] Mumford J, Atkinson K, Bench-Capon T. Explaining Factor Ascription. In: Proceedings of JURIX 2021. IOS Press; 2021. p. 191-6.

[7] Al-Abdulkarim L, Atkinson K, Bench-Capon T, Whittle S, Williams R, Wolfenden C. Noise induced hearing loss: Building an application using the ANGELIC methodology. Argument and Computation. 2019;10(1):5-22.

[8] Al-Abdulkarim L, Atkinson K, Bench-Capon T. A methodology for designing systems to reason with legal cases using ADFs. Artificial Intelligence and Law. 2016;24(1):1-49.

[9] Bench-Capon T. Using Issues to Explain Legal Decisions. In: EXplainable and Responsible AI and Law 2021. vol. 3168. CEUR Workshop Proceedings; 2021. .

[10] Brewka G, Woltran S. Abstract Dialectical Frameworks. In: 12th International Conference on the Principles of Knowledge Representation and Reasoning; 2010. .

[11] Gordon TF, Prakken H, Walton D. The Carneades model of argument and burden of proof. Artificial Intelligence. 2007;171(10-15):875-96.

[12] Gordon TF. Visualizing Carneades Argument Graphs. Law, Probability and Risk. 2007;6(1-4):109-17.

[13] Gordon TF, Walton D. Formalizing Balancing Arguments. In: Proceedings of COMMA 2016; 2016. p. 327-38.

[14] Gordon TF. Defining argument weighing functions. IfCoLog Journal of Logics and their Applications. 2018;5(3):747-73.

[15] Frühwirth T. Constraint Handling Rules. Cambridge University Press; 2009.

[16] Gordon TF, Friederich H, Walton D. Representing Argumentation Schemes with Constraint Handling Rules (CHR). Argument and Computation. 2017;9(2):91-119.

[17] Horty JF. Reasoning with Dimensions and Magnitudes. In: Proceedings of the 16th ICAIL; 2017. p. 109-18.

[18] Bench-Capon T. Explaining legal decisions using IRAC. In: Computational Models of Natural Argument 2020. vol. 2669. CEUR Workshop Proceedings; 2020. p. 74-83.

[19] Bench-Capon T, Atkinson K. Implementing factors with magnitude. In: Computational Models of Argument. IOS Press; 2018. p. 449-50.

[20] Atkinson K, Collenette J, Bench-Capon T, Dzehtsiarou K. Practical tools from formal models: the ECHR as a case study. In: Proceedings of the 18th ICAIL; 2021. p. 170-4.