What Makes a System a Legal Expert?

Trevor BENCH-CAPON [1],
Department of Computer Science, University of Liverpool, UK

Abstract. Since the 1980s, AI and Law has attempted to capture legal expertise in computer programs. But what is this expertise? This paper reviews a number of approaches, from the 1980s to the present day, which represent different answers to this question. It argues that our notion, and understanding, of expertise has developed and improved over the decades. As yet, however, only a few rather specific aspects have been addressed in detail, in particular the move from intermediate predicates to legal consequences, and the distinguishing of precedents. Much more, including the moves from evidence to facts and from facts to intermediate predicates, awaits exploration.

Keywords. expert systems, legal reasoning, legal expertise, argumentation schemes

1. Introduction

2012 sees the twenty-fifth Jurix conference, and so provides a good opportunity for some retrospective consideration of where we have come since the 1980s. In this paper I will look at our evolving understanding of what is needed to put the 'expert' in a legal expert system. One *caveat*: I am a computer scientist, not a lawyer, and so my own notion of legal expertise is heavily influenced by what I have seen represented in AI and Law systems. This is the perspective from which I discuss the expertise included in these systems: if lawyers see some things as missing, or different, the paper will at least provide a clear target for their criticisms, and perhaps the future practice of computer scientists can be modified accordingly.

2. Formalisation of Legislation

For our first example of an attempt to represent legal expertise, we may consider the classic British Nationality Act System of Imperial College [15]. The idea here is that the question *what do lawyers know?* can be answered simply with *lawyers know the law*. Thus the system comprises little more than a set of Horn Clauses intended to capture the law as stated in the British Nationality Act 1981 as a logical theory. Then, given a set of facts, using a standard reasoning procedure, such as classical deduction, the program can derive the consequences of the theory: in particular when augmented by a set of relevant facts regarding a person (their age, place and date of birth, parentage and the like) it can derive an answer to the key question of whether the person concerned is

[1]Corresponding Author: Trevor Bench-Capon; E-mail: tbc@liverpool.ac.uk

a British citizen[2]. Gathering information was similarly general, using the APES expert system shell which employed the Query-the-User technique [14]: some predicates (the facts relating to particular people) were declared *askable*, and if they became relevant the user would be asked if they were true.

In fact this approach yielded a remarkably impressive system in the case of the British Nationality Act, and the system inspired a number of imitators, e.g. [16] and [7]. These, however, were far less convincing than the original. What was very important to the success of the BNA system was that the questions asked - age, place and date of birth and the like - were all straightforward questions, readily understood by the layman, and they sought information that almost everyone knows about themselves. But this is not invariably the case, and in other areas the questions that come to the user are abstruse and technical and not immediately answerable. Also, very often the questions that come from the legislation are vague or even ambiguous. The authors of [15] write:

> "In addition to vagueness, legislation is generally thought to contain both imprecision and ambiguity. ... In fact, we found fewer such examples than we originally expected. In practice, where imprecision or ambiguity did exist, it was usually possible to identify the intended interpretation with little difficulty."

.

This may have been true of the BNA (or possibly elements of vagueness and ambiguity went unnoticed by the authors of [15]): if so it was a somewhat untypical piece of legislation and this finding did not generalise very far. In the next section we will consider US Trade Secrets law, the domain of HYPO [2], CATO [1] and IBP [9], to explore these issues further.

3. BNA Approach Applied to US Trade Secrets Law

Issue-Based prediction (IBP) [9] uses the Restatement of Torts to provide a top level 'logical model' based on the Restatement of Torts. Although the Restatement is not strictly speaking legislation, it plays the same role and if the domain were governed by statutory provisions we would expect them to have a similar import: a similar top layer of logical rules for Home Office Deduction was used in the CABARET system [17].

The top level of IBP, written as a logic program[3], is:

```
tradeSecretMisappropriation(X):-infoTradeSecret(X),
                                infoMisappropriated(X).
infoTradeSecret(X):-informationValuable(X),
                    effortsToMaintainSecrecy(X).
infoMisappropriated(X):-infoUsed(X),
                        confidentialRelationship(X).
infoMisappropriated(X):-improperMeans(X).
```

Figure 1: Trade Secrets a la BNA

[2]Strictly [15] could derive the answer if the facts gave a positive answer. Otherwise, it answered 'no': inability to demonstrate that someone was a citizen was taken as a reason to say that they were not a citizen.

[3]I use Prolog to present program code.

If we were writing a US Trade Secrets law system in the style of [15], this (together with declarations marking the predicates in the bodies of the last three clauses as *askable* and supplying suitable text prompts) would be the whole program. Running with Query of the User, the user would need to answer a sequence of questions:

- Is the Information valuable?
- Were Efforts taken to maintain secrecy?

Answering *yes* to both these questions would establish that the information was a Trade Secret and lead to the next set of questions, while answering *no* to either would mean a finding for the defendant.

- Was the Information used?
- Were the Plaintiff and the Defendant in a Confidential relationship?

Answering *yes* to both these questions would establish that the information was misappropriated, finding for the plaintiff. Answering no to either would lead to

- Were Improper Means used to Obtain the Information?

Again answering *yes* would establish that the information was misappropriated and lead to a finding for the plaintiff. Answering *no* would mean a finding for the defendant.

This shows up fairly clearly why this approach will not work in all domains. The questions asked of the user are simply not appropriate for a lay person. To answer, for example, whether the plaintiff and the defendant were in a confidential relationship, one needs to know a good deal about what counts as a confidential relationship and this will involve knowing a good range of previous decisions, not confined to Trade Secrets law. Indeed the rationale for systems such as HYPO and CATO is to provide the case-based knowledge to answer just these questions, which are recognised as requiring considerable legal expertise to answer.

Seen in this light, the formalisation of the legislation tells us not what the law is - for that we need the elaboration in terms of cases - but rather the *problem solving* knowledge of the system: the questions that need to be answered, the points at which the plaintiff's case may be attacked. The formalisation also determines the order in which they are answered. That this may be a problem was recognised in the BNA program:

> The quality of this interactive dialogue is sensitive to the order in which the different rules for acquiring citizenship are written, and to the order of the conditions within individual rules. [15], p371.

Ordering is, however, not a problem where the questions are posed internally as they are in IBP and CABARET. In both these programs the division into issues by providing a set of questions to answer provides a very useful structure which can then be filled out with the case law.

4. Including Case Law

Thus far we have seen that knowledge of the legislation is by no means enough: it is useful in that it tells us which questions must be answered, and the consequences of the answers given, but it provides us with very little assistance in answering them. And it

is this that we would expect an expert lawyer to assist us in. So what do the case-based systems add to capture this aspect of expertise?

The case-based systems provide two things: a collection of cases, together with their outcomes, and a means of describing cases (*factors* in CATO and IBP, focal slots and *dimensions* in HYPO).

The usage of the term 'factor' varies somewhat in different descriptions of HYPO and in CATO and is discussed in [13]. A dimension can be seen as a consideration relevant to decisions in the domain of interest. For example, in US Trade Secrets Law, two dimensions would be whether the secret had been disclosed (*Secrets-Disclosed-Outsiders*), or whether a common employee had been paid to change employers (*Bribe-Employee*). Dimensions may be applicable or not applicable, but if applicable they have a range (the number of people to whom the secret had been disclosed, or the size and nature of the bribe). At one end of the range the plaintiff is favoured (e.g. no disclosures), at the other the defendant is favoured. Where the facts put the case somewhere in the middle of the range, both sides are more or less favoured. Factors can be seen as points on the dimension: they are present or absent in a case. If present they fixedly favour one side or the other. Often a dimension maps into two factors, one representing one extreme and the other the entirety of the rest of the range. For example, *no bribe* favours the defendant, while any bribe at all favours the defendant. Sometimes a dimension maps into several factors. Thus voluntary disclosures may either be disclosure to specific outsiders or disclosure in a public forum, both of which favour the defendant, but clearly the latter favours the defendant more strongly. In HYPO the applicability of dimensions was supposed to be determined by the facts of the case: in CATO cases were simply supplied to the program already represented as collections of factors.

Thus a key piece of expertise would seem to be knowledge of how to describe the cases. The facts of the cases differ greatly, but if they are to be compared and act as precedents for one another they need to be described on some common basis. One well known set of cases in AI and Law require us to be able to subsume descriptions of chasing a fox with horse and hound, waiting to shoot a flock of passing ducks, trawling for a shoal of fish and attempting to catch a baseball under a common description (the wild animals cases and *Popov v Hayashi*, described in AI and Law 20(1) [3]). The additional knowledge thus includes the relevant considerations (whether dimensions or factors) and how they relate to the case outcomes. Ideally we would want to know how they relate to facts, but this is not very explicit in any of these systems, and entirely absent from CATO and IBP. We will need to return to this point below.

As well as simplifying dimensions into factors, CATO introduced the notion of a *factor hierarchy*. The factors describing the case (the *base level* factors) are children of more abstract factors. Where the abstract factor is pro-plaintiff, its pro-plaintiff children suggest that it is present and its pro-defendant children suggest that it is absent. Where the abstract factor is pro-defendant, the reverse is, of course, the case. The abstract factors are in turn the children of more abstract factors (issues, essentially the predicates found in the Prolog program of Figure 1).

Viewed in one way, the factors could play a very similar role to the way case law is used to extend a BNA style program, as described in, e.g. [4], in which certain fact patterns are held, on the basis, of case law, to provide sufficient conditions for (or against) a predicate found in legislation to hold. For example we could add to the program from the last section the following procedure to determine whether `improperMeans(X)`

were used. The first three clauses provide sufficient conditions for the predicate to hold, the fourth a sufficient condition for it to fail.

```
improperMeans(X):-bribeEmployee(X).
improperMeans(X):-deception(X).
improperMeans(X):-invasiveMethods(X).
improperMeans(X):-reverseEngineered.(X),!,fail.
```

Figure 2: Improper Means with Factors as Sufficient Conditions

However, this embodies a great deal more than can be justified on the basis of the CATO representation. While CATO identifies the four considerations, this treatment:

- decides that these factors represent *sufficient* conditions;
- imposes a priority ordering: if two of the conditions hold it is the first encountered in the program that will be applied. Thus in Figure 2 `reverseEngineered` is the least important and will not be considered if any of the other three factors apply. But it could be written as the first clause, in which case it would be decisive if present, or in some intermediate position.
- imposes the burden of proof on the plaintiff: if none of the factors apply `improperMeans` will be taken to be false. The opposite burden of proof could be given by adding `improperMeans(X)` as the last clause of the procedure.

Although this approach has been applied in the logic programming tradition, it is hard to justify to lawyers. It lacks any real expertise in *using* the factors: in a very real sense it embodies the thinking of a computer scientist rather than the thinking of a lawyer.

One way of expressing dissatisfaction with the approach of Figure 2 is that it does not treat the case as a whole, and so the context is lost. An approach using a rule based representation which does preserve the integrity of the case was developed in [12]. Here every case was represented as three rules:

- a rule with all the pro-plantiff factors present as antecedent and a decision for the plaintiff as consequent;
- a rule with all the pro-defendant factors present as antecedent and a decision for the defendant as consequent;
- a rule expressing the priority between these rules, according to the actual outcome of the case.

This can, of course, be very easily expressed as a Prolog program. Suppose we have a case with pro-plaintiff factors `bribed-employee` and `unique-product` and pro-defendant factor `reverse-engineered`, found for the plaintiff and another case with pro-plaintiff factors `security-measures` and `unique-product` and pro-defendant factor `reverse-engineered`, found for the defendant. This produces the following program (the order of the clauses captures the priorities):

```
winner(p):- bribed-employee, unique-product.
winner(d):- reverse-engineered.
winner(p):- unique-product, security-measures.
winner(d).
```

Figure 3: Trade Secrets a la Prakken and Sartor 88

Figure 3 presents a faithful encapsulation of the knowledge of the two cases, with the additional information that the burden of proof is on the plaintiff. This could be altered by changing the final clause to `winner(p)`. The drawback with the program of Figure 3 is that it is a faithful *encapsulation* of the knowledge to be gained from the two cases, but it lacks any *expertise*: it cannot use this knowledge to go beyond what is given there. Reasoning is strictly *a fortiori*, so that given a new case with pro-plaintiff factors `bribed-employee` and `security-measures` and a pro-defendant factor `reverse-engineered`, we would have to find for the defendant, on the basis of the second clause, whereas the plaintiff's lawyer ought to be able to make a plausible argument for her client. Indeed we would might expect (if the case was one of our precedents rather than the new case), that the case would have been decided for the plaintiff, so that `bribed-employee` and `security-measures` would represent the most important sufficient factor.

Something of the required expertise can be found in [18], which provides some assistance in how to use cases by giving a set of argument moves with which to deploy the cases. Among these are moves to use when one has no exact match with a favourable precedent, but one does have a near miss, which can be adapted for use in an argument. Even so, thinking in terms of sets of factors, rules and priorities does savour more of scientific than legal thinking. The point is that when people go to Law School they do not go primarily to learn information, neither statutes (which would be rather transitory knowledge in any case), nor cases. While they do consider cases, the point is not to learn large quantities of them. The best lawyer is not the one who knows the most cases: provided there is access to a decent library, or on-line facility, this knowledge is not essential. It is not knowing the cases themselves but knowing what can be *done* with them them, how arguments for particular positions can built from them, that is the goal of legal education. The aim is to acquire certain habits of thought, to come to *think like a lawyer*. So let us look again at the classic case-based systems from this perspective.

5. Use of Cases in HYPO and CATO

If we consider the way in which HYPO uses its cases, we see immediately that it deploys them in the context of what it calls *three-ply argumentation*. This framework firstly presupposes an adversarial setting, with the first and third plies being supplied by one party (we will here assume the plaintiff's counsel) and the second by her opponent (here counsel for the defence). The three plies are:

- *Cite a Case*. Here the counsel chooses some precedent with the desired outcome to cite as reason to find for their side. The choice is not, of course, arbitrary, but should represent the best precedent. The best precedent is determined by HYPO by forming a lattice of cases according to how 'on-point' the cases are, with on-pointness calculated in terms of their shared dimensions.
- *Dispute Precedent*. In this ply the opposing counsel puts forward reasons why the precedent cited in the first ply should not be applied. There are two ways to do this: offering a counter example or distinguishing the precedent. A counter example is a precedent found for the defendant which is at least as on-point as the precedent cited in the first ply. Distinguishing involves finding a difference between the current case and the precedent cited which either makes the precedent

stronger that the current case for for the plaintiff, or the current case stronger for the defence.

- *Rebuttal*. Finally the plaintiff will have the opportunity to rebut the arguments from the second ply. Most important here is the ability to distinguish any counter examples, but the plaintiff's counsel may also cite additional precedents to show that any weaknesses identified in the second ply are not fatal (e.g. if a case was distinguished by the absence of some feature present in the precedent, cases lacking the feature found for the plaintiff could be cited), or to emphasise strengths (e.g. pointing to features strengthening the current case for the plaintiff but not present in the precedent).

This encapsulates a view of what it means to approach a case like a lawyer:

- Awareness that the setting is intrinsically adversarial. The counsel do not take a dispassionate or balanced view: they are supposed to find and advance arguments only for their side.
- Understanding of the notion of on-pointness. This is a rather different notion from the kind of triangular distance or least squares matching that we might find in a mathematical application. Notice also that, in HYPO, this is done in terms of dimensions, rather than the points on them.The relevance of different locations on dimensions is that a difference in position might provide a way of distinguishing the cases in the next ply.
- Understanding what counts as a counter example.
- Understanding of what can (and what cannot) be used to distinguish a case.
- The kind of supporting information that can be deployed in the rebuttal.

Much of this is now very familiar to people working in AI and Law, but it is not obvious that this is the way to go about arguing with cases. An approach more akin to multiple linear regression, Bayes nets or some other statistical approach such as using neural networks might have seemed (to a computer scientist) the natural way to go if one had chosen to ignore the ways lawyers reason in practice, and simply to apply standard AI techniques to the problem (e.g. [6]).

While HYPO was targetted quite generally at legal practioners seeking arguments based on precedent cases to support their clients, CATO was specifically aimed at law *students*. Moreover it was designed to teach these students a specific skill, namely how to distinguish a precedent, and what makes a distinction a good distinction. Remember, not every difference can be used to distinguish a case. To distinguish a case cited for the plaintiff, one must find a difference which strengthens the current case relative to the precedent for the defendant, or which weakens the current case relative to the precedent for the plaintiff. The differences between CATO and HYPO are instructive because they show what was was considered vital to the teaching of this particular task, to inculcating this specific piece of legal expertise.

First CATO dispenses with the notion of dimensions altogether and represents cases solely as bundles of factors, so abstracting away all factual issues. This means that the students can focus single-mindedly on the differences in terms of the abstract, legally relevant, considerations represented by factors. Moreover, CATO introduces the notion of abstract factors and the factor hierarchy. The role of the factor hierarchy is to enable moves evaluating and responding to distinctions to be made. If we have a factor which has siblings favouring the same side (e.g `bribedEmployee`

and deception are both children of ImproperMeans favouring the plaintiff) one of which is present in the current case and the other in the precedent, it is possible to *downplay* the distinction. If it is argued that a case can be distinguished because the precedent contains deception, while deception is missing from the current case, we can respond that bribedEmployee is present in the current case, so that ImproperMeans were still used, and so the cases cannot be distinguished after all. Similarly if the current case or the precedent contains siblings which argue for both the presence and absence of the parent, downplaying is possible. Thus, if the precedent contains nothing relating to Impropermeans but the current case contains both deception and reverseEngineered, an attempt to argue that deception distinguishes the current case in favour of the plaintiff, can be met by a claim that, since reverseEngineered is also present, it is not clear that the trade secret was obtained by ImproperMeans. Distinctions which can be downplayed are, of course, to be avoided if possible, since they can be rendered inconclusive or even neutralised.

In contrast if a distinction cannot be downplayed it is more desirable, and it is possible to emphasise its strengths by pointing to the abstract factor that is present (or absent) and explicitly stating that this was not so with the precedent.

6. Representing Expertise Declaratively

So far we have argued that the expertise that a good lawyer has is less a matter of knowledge, whether knowledge that or knowledge how, but rather certain habits of thought, particular ways of approaching a problem, and recognising the ways in which information needs to be used. All this is largely absent from programs built using rules, but is present in the case-based systems. It is, however, present in these systems only in the code itself, or in discussions about the system: there is no explicit representation of the expertise. And it was just this declarative capture of expertise that was the original goal of expert systems. So, can this expertise be made explicit and declarative? Argumentation schemes are currently are an important topic in research on computational argumentation, and they may provide the answer, as explained below.

Typically argumentation schemes are understood to offer a collection of stereotypical patterns of reasoning permitting conclusions to be presumptively drawn, assuming certain felicity conditions (often represented by critical questions) are not violated. Expert Opinion is often taken as the paradigm of such argumentation schemes. Analysis of the Supreme Court opinions in *Furman v Georgia* [5] showed a number of these. But also, in Brennan's opinion the analysis identified something more interesting:

> The four principles in his test are substantially equivalent to four dimensions which can favour the plaintiff in these cases. His cumulative test therefore could be seen as an application of the HYPO method to the current case, assuming that an appropriate analysis of the precedents existed. In particular the way he uses the test once found, with none of the principles being necessary or sufficient, individually or collectively, and all taken as established to a greater or lesser extent. is very much in accord with this approach. In the course of his opinion he effectively analyses the case in the same way as a knowledge engineer building a HYPO system for this domain would.

In fact argumentation schemes do not always represent a way of moving premises to conclusions, but are sometimes better seen, e.g. [11], as a reasoning method in them-

selves. In other words, some schemes capture not a sterotypical pattern of inference, but a method for reasoning, exactly what we have come to understand as constituting legal expertise and exemplified in Brennan's opinion in *Furman*. Now consider the argumentation schemes of [19], further developed in [20]. There the reasoning of CATO is reconstructed as a set of argumentation schemes, which are highly related in that there is a main scheme and the remainder of the schemes are used either to establish the premises needed by another scheme in the set, or to provide undercutters for a scheme in the set. Taken together the schemes provide a cascade of arguments which follow the three-ply structure of HYPO and CATO, and so using the schemes will result in following the CATO reasoning method; with the techniques for making pertinent distinctions, downplaying them and emphasising them all expressed as particular schemes. Thus these schemes can be seen as explicit, declarative, representations of the methods that CATO is attempting to instill in its students. But this is only the tip of the iceberg. Legal reasoning can be seen (e.g. [10]) as a two step process of reasoning from facts to intermediate predicates, and then from intermediate predicates to legal consequences. Factors can be seen the intermediate predicates. But whereas this second step is relatively tractable, because it does not require knowledge about the world, the first step is deeply problematic (e.g. [8]), because of the amount and diversity of knowledge required. And even then they may be further steps required, such as a step to establish the facts of the case on the basis of the available evidence, requiring yet more knowledge, and specialised argument schemes.

7. Concluding Remarks

The schemes of [19] and [20] are specific to reconstructing the reasoning covered by CATO. They represent a technique for distinguishing cases, and so do not cover several other aspects of legal reasoning with cases. They begin with factors and so do not address any questions of how factors are identified in the first place, how they are assigned to cases, or how it is determined which side they favour. All these are questions for which expertise - thinking like a lawyer - is required. Indeed much of a lawyer's skill must be seen as lying precisely in deciding, given a set of facts, how best to present these facts to support the assignment to the case of a set of factors favouring her client, and factors which will align the case with favourable precedents. Finding sets of argumentation schemes which correspond to the ways lawyers tackle these questions, and the objections that can be made to counter unfavourable proposals, would, I suggest, be a highly fruitful way of attempting to pin down the required expertise. Unlike distinguishing, however, there is no existing AI and law program which embodies this method, and so there are not the same firm foundations from which to develop such a set of schemes. Attacking this problem is therefore a substantial challenge for the future.

References

[1] V. Aleven. *Teaching Case Based Argumentation Through an Example and Models*. Phd thesis, University of Pittsburgh, Pittsburgh, PA, USA, 1997.

[2] K. D. Ashley. *Modeling Legal Argument*. MIT Press, Cambridge, MA, USA, 1990.

[3] Katie Atkinson. Introduction to special issue on modelling popov v. hayashi. *Artif. Intell. Law*, 20(1):1–14, 2012.

[4] Katie Atkinson and Trevor J. M. Bench-Capon. Legal case-based reasoning as practical reasoning. *Artif. Intell. Law*, 13(1):93–131, 2005.

[5] T. Bench-Capon. Towards computational modelling of supreme court opinions. In K. Atkinson, editor, *Modelling Legal Cases*, pages 77–90. Huygens Editorial, Barcelona, 2009.

[6] Trevor J. M. Bench-Capon. Neural networks and open texture. In *Proceeding of the 4th International Conference on AI and Law*, pages 292–297, 1993.

[7] Trevor J. M. Bench-Capon, G. O. Robinson, Tom Routen, and Marek J. Sergot. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *Proceedingts of the First International Conference on AI and Law*, pages 190–198, 1987.

[8] Joost Breuker and Nienke den Haan. Separating world and regulation knowledge: Where is the logic. In *proceedings of the 3rd International Conference on AI and Law*, pages 92–97, 1991.

[9] Stefanie Brüninghaus and Kevin D. Ashley. Predicting outcomes of case-based legal arguments. In *Proceedings of the Ninth International Conference on AI and Law*, pages 233–242, 2003.

[10] Lars Lindahl and Jan Odelstad. Open and closed intermediaries in normative systems. In *Proceedings of JURIX 2006*, pages 91–99, 2006.

[11] H. Prakken. On the nature of argument schemes. In C.A. Reed and C. Tindale, editors, *Dialectics, Dialogue and Argumentation*, pages 167–75. College Publications, 2010.

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

[13] Edwina L. Rissland and Kevin D. Ashley. A note on dimensions and factors. *Artif. Intell. Law*, 10(1-3):65–77, 2002.

[14] M. Sergot. A query the user facitiy for logic programs. In M. Yazdani, editor, *New Horizons in Educational Computing*, pages 145–163. Ellis Horwood, 1984.

[15] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, F. Kriwaczek, Peter Hammond, and H. T. Cory. The british nationality act as a logic program. *Commun. ACM*, 29(5):370–386, 1986.

[16] D. M. Sherman. A prolog model of the income tax act of canada. In *Proceedingts of the First International Conference on AI and Law*, pages 127–136, 1987.

[17] David B. Skalak and Edwina L. Rissland. Argument moves in a rule-guided domain. In *Proceedingts of the Third International Conference on AI and Law*, pages 1–11, 1991.

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

[19] Adam Wyner and Trevor Bench-Capon. Argument schemes for legal case-based reasoning. In *Proceedings of JURIX 2007*, pages 139–149, 2007.

[20] Adam Wyner, Trevor Bench-Capon, and Katie Atkinson. Towards formalising argumentation about legal cases. In *Proceedings of the 13th International Conference on AI and Law*, pages 1–10, 2011.