Arguing With Cases

T.J.M. Bench-Capon

LIAL - Legal Informatics at Liverpool Department of Computer Science The University of Liverpool Liverpool England

In this paper I attempt to consolidate work in AI and Law concerning reasoning with cases. I distinguish reasoning with cases from retrieving cases, and from reasoning with information derived from cases. I then present an algorithm for generating legal arguments drawing on work done in two of the most successful programs for reasoning with cases in AI and Law, HYPO and CATO. I conclude by drawing attention to three areas of possible future investigation.

1. Introduction

Case based reasoning (CBR) is a central element in AI and Law. Reasoning with cases is a distinctive and well established part of legal reasoning, and CBR is a prime example of where AI and Law work has made a significant contribution to AI in general. The leading examples of this work derive from the work on HYPO, developed by Rissland and Ashley and most completely described in (Ashley 1990), and subsequently CABARET (Skalak and Rissland 1991), and CATO (Aleven 1997).

In this paper I shall attempt to take stock of CBR in AI and Law. I shall begin by distinguishing CBR proper from some other uses of cases in AI and Law, and then briefly describe a recent attempt to assimilate CBR into a rule based framework (Prakken and Sartor 1997). The bulk of the paper will give an algorithmic description of how to construct a legal argument using cases. This will draw heavily on HYPO and CATO, but will extend HYPO by using the notion of a factor hierarchy introduced in CATO, and extend CATO by using dimensions, prominent in HYPO, but omitted from CATO.

2. Case Based Reasoning versus Case Retrieval

It is important to distinguish at the outset CBR from another very common use of cases in legal information systems, such as those described in (Kowalski 1991), and Montazeri et al (1997). In those systems the idea is, given a new case, to use a matching process to identify the closest case or cases. Once this case has been identified, the principle of stare decisis is applied to ascribe the decision in the closest case to the new case. Such systems are interesting, and in particular the determination of similarity between cases is a worthwhile study, but the reasoning (at least as done by the system rather than the user) with cases is limited to simply using stare decisis. This is a gross simplification, and fails entirely to reflect the subtleties of CBR proper as modelled in HYPO and CATO. Moreover the decisions that emerge are very prone to error, since the determination of the closest cases is rather hard to capture in a general way. Such systems are better regarded not as CBR at all, but rather as case retrieval systems, and better evaluated using the standard Information Retrieval measures of relevance and recall rather than the accuracy of their decisions.

In CBR proper, the output is not simply a case to follow but an argument. This in turn means that one of the inputs to the CBR system is the side to argue for. Also, because we are interested in producing an argument, the cases we wish to find and deploy are not determined by their similarity alone, but rather by the role that they can play at given points in the argument. For this reason we are as interested in the nature of the differences as much as their extent .
To summarise, CBR, unlike case retrieval requires:

1. A position to argue for
2. A structure for a case based argument, determining a variety of argument moves
3. Consideration of cases with reference to the argument moves they s upport

In passing we may remark that approaches based on the statistical and machine learning or neural net analysis of cases (e.g Stranieri and Zeleznikow 1997), suffer from similar defects. They do not yield an argument, and again rely entirely on stare decisis.

3. Terminology

Before discussing a rule based approach I shall introduce some terminology. First a case is described in terms of the factors it exhibits. Essentially factors are legal issues which arise in the domain; from the domain of trade secrets, used in Ashley (1990), we can instance such things as whether a specific nondisclosure agreement existed, and whether the product was reverse engineerable. Each factor will either favour the plaintiff or the defendant (which I will call p-factors and d-factors, respectively).
Some factors will be binary, "yes or no": others may be quantifiable, such as whether a competitive advantage was gained, which can be measured in time or money, and others will be enumerable, and capable of having an order placed on their possible values, such as whether security measures were taken which can range from minimal measures to individual employee nondisclosure agreements. In the non-binary cases the factors will have a strength and direction as well as simply applying or not. For example taking security measures favours the plaintiff, and the more draconian the measures the more the plaintiff is favoured. I shall, following HYPO, call these factors dimensions. Thus factors comprise binary factors and dimensions. A case will typically exhibit a number of factors. In CATO some 26 factors applicable to trade secrets law have been identified.

Existing precedent cases are analysed according to their factors, and new cases must similarly have their factors identified. Note that these factors are not the simple observable facts of the case, but must be identified, either by hand, or by using some set of rules to map facts into factors, or perhaps eventually automatically from the free text description of a case (Bruninghaus and Ashley 1997). However it is done, cases must be associated with their factors before CBR commences.

A case also has an outcome, which may be a finding for the plaintiff or for the defendant. I shall refer to cases as p-cases and d-cases according to whether the finding was for the plaintiff or for the defendant.
Suppose we wish to argue for the plaintiff. If we compare two cases, C1 and C2, the set of factors exhibited by the cases may be classified into four groups:

A. p-factors common to the two cases
B. d-factors common to the two cases
C. factors which make C1 stronger for the plaintiff than C2: (i.e. p-factors in C1 but not in C2 and d-factors in C2 but not in C1)

D. factors which make C1 weaker for the plaintiff than C2: (i.e p-factors in C2 but not in C1 and d-factors in C2 but not in C1).

If we have factors in all these classes we say that C2 is an ABCD-case with respect to C1; or if factors in A and D were present C2 would be an AD-case with respect to C1. Note that ABCD-cases are intended to include all AD-cases, etc.

As will be seen later it is this classification of the cases, rather than any notion of closeness, which determines how we can use C2 in an argument concerning C1.

4. A Rule Based Reconstruction of HYPO/CATO

In Prakken and Sartor (1997) the authors attempt to reconstruct the reasoning of a HYPO/CATO style system within their logical based argumentation framework. In their treatment, cases are represented as follows:

1. A rule which conjoins the p-factors in the case as the antecedent with p-outcome as consequent;
2. A rule which conjoins the d-factors in the case as the antecedent with d-outcome as the consequent;
3. A priority relation which states that (ii) < (i) in a p-case and (i) < (ii) in a d-case.

This representation gives a set of rules which can be used to construct arguments for either side, according to the scheme given in Prakken and Sartor (1996). Because a new case may not match any of the rules exactly, they permit the rules to be broadened, that is invoked even if the new case satisfies only some of the factors in the antecedent. They also permit a rule to be distinguished; given a rule
factor(1), ... factor(n) -> outcome
which has been broadened by omitting one or more of the factors the distinguishing rule is
naf factor(i) -> neg outcome,
where factor(i) is any of factor(1) ... factor(n) not in the broadened rule, naf is weak negation and neg is strong negation.

They also define a dialogue game in which these rules may be deployed. For a full account see Prakken and Sartor (1997).

This paper is a valuable extension to their argumentation framework, since it allows cases to be used as a source of premises for arguments, in addition to the legislation derived premises they used previously. Also it does capture much of the underlying logic of the reasoning used in HYPO/CATO. However, considered as an account of CBR it is rather less satisfactory. Although cases are used to supply the pool of premises, they do not exist as cases. Thus the choice of the rule to broaden does not depend on other factors that may be present in the case and the precedent; moreover since any broadened rule may be distinguished, the strategy will in general be likely to fail. Overall the account is one of reasoning with information derived from cases rather than with cases themselves. This is important: the rhetorical force of a counterexample derives not from the rule it presents, but from the other factors present in the case. Finally, although the approach models four of the eight argument moves listed in CATO (Aleven and Ashley 1997), there are four other moves which have no place in Prakken and Sartor's system.

Attractive and illuminating as their approach is, therefore, we cannot concede that it represents a model of reasoning with cases.

5. Arguments in CATO and HYPO

In CATO there are eight argument moves:
C1 Analogising a case to a past case with a favourable outcome;
C2 Distinguishing a case with an unfavourable outcome;
C3 Downplaying the significance of a decision;
C4 Emphasising the significance of a distinction;
C5 Citing a favourable case to emphasise strengths;
C6 Citing a favourable case to argue that weaknesses are not fatal;
C7 Citing a more on point counterexample to a case cited by an opponent;
C8 Citing an as on point counter example to a case cited by an opponent.

Note that all of these moves are characterised in terms of their rhetorical effect. CATO has been implemented as a training tool, and these moves are solicited from students in the course of a mini-dialogue with the system about a case.

HYPO structured its arguments around a 3-ply form. Corresponding CATO arguments are given in brackets:

1. A point is made for one side (C1, C5, C6);
2. A response to this point is made on behalf of the other side; it may be a counter example (C7 or C8), or a distinction (C2);
3. A rebuttal of the response is made on behalf on the original side; essentially this involves distinguishing any counter examples (C2).

Note that CATO has two more argument moves than were used in HYPO. Note also that these are concerned not with deduction, but with making the case more persuasive. In fact HYPO could not produce these moves since they rely on the notion of a factor hierarchy. The factor hierarchy has the factors as its leaves. The parents of factors are known as abstract factors, and are related to their children either in a positive or negative way, according as to whether the factor promotes or militates against the abstract factor. There may be several layers of abstract factors, until parentless nodes are reached. These are known as issues.

In our algorithmic description we will follow the argument structure of HYPO, but use CATO's argument moves, and additionally allow moves C5 and C6 to be made as part of the rebuttal. Note that HYPO also modelled various strategies for posing hypothetical variations of a problem which would strengthen or weaken the argument of a particular side. This aspect of HYPO is not summarised here. See Ashley (1990) pp 147-54 for a discussion of this aspect.

6. An Algorithmic Description of Case Based Argument

In order to simplify the following description, I shall assume that we are arguing for the plaintiff. The algorithm would equally well find arguments for the defendant, if we make the appropriate substitutions. Assume therefore that we have a new case (NC), and we wish to argue for the plaintiff.

An argument will comprise a number of three ply arguments. Thus

Argument Algorithm (AA)

AA1 Find all citeable favourable cases
AA2 Until no response possible or no more citeable cases:
    Construct 3-Ply argu ment for citeable case
    Next citeable case
AA3 End

A diagrammatic view of the algorithm is given in Figure 1.

Figure1: Argument Algorithm

A citeable case is one which can form the basis of moves C1. The cases should be cited roughly in the order of the best case first.

In order to form the basis of a move C1, a case must be a p-case, and an A-case with respect to NC. The precedent is better if it is also a B-case and/or a C-Case with respect to NC and worse if it a D-case with respect to NC.

Thus to carry out step A1:

Find Citeable Cases Algorithm (CC)

For all p-cases in case base, with respect to NC;

CC1 Find all A cases
CC2 Order A-Cases as follows:
AC, ABC, AB, A, ACD, ABCD, ABD, AD;

It is likely that not all the above types of case will be found; for example to be an AC case it is necessary that there are no pro-defendant factors in NC, which makes it rather trivial.

We now consider the three ply argument:

3-Ply algorithm (TP)

TP1 State point
TP2 Respond
TP3 Make rebuttal

Stating the point is simply to list the p-factors in the case, to make the claim and to cite the case.

State point algorithm (SP)

SP1 Write "Court should find for the plaintiff where:"
SP2 For p-factor in NC and precedent:
Write p-factor
SP3 Write "Although"
SP4 for d-factor in NC and precedent
Write d-factor

SP3 Write "Cite:"
SP4 Write citation for precedent

To respond to this point, the defendant can use, in order of strength, moves C7, C8 or C2. Thus:

Respond Algorithm (RA)

RA1 Find counter examples
RA2 State counter examples
RA3 State factor distinctions and
Emphasise significance of the distinction where appropriate
RA4 Find dimension distinctions
RA5 State dimension distinctions

A precedent is a counter example if it is a d-case. A counter example is essentially a citeable case for the opposing side. A counter-example is as on-point if has all the A-factors and B-factors of the cited case. A case is more on-point if it contains additional A and B factors.

Counter Example Algorithm (CE)

For all d-cases in case base, with respect to NC,

CE1 Find all A cases
CE2 Remove cases where there exists an
A-factor in cited case not in case;
CE3 Remove all case where there exists a
B-factor in cited case not in case
CE4 Label cases where there is an A-factor
or a B-factor in case not in cited case "mop";
CE5 Return list of remaining cases as counter_examples

To state the counterexamples we must cite the case, say whether it is more on-point, and if so draw attention to the factors which make it so.

State counter examples Algorithm (SC)

SC1 Write "Counter examples"
SC2 For case in counter_examples
Write citation for case
If mop case
Then write "is more on point and held for
defendant where it was also the case that"
Else write "is as on point and held for the
defendant"
GOTO SC2
SC3 For A-factor in case not in cited case
Write A-factor
SC4 For B-factor in case not in cited case
Write B-factor

We can distinguish a case in one of two ways, either on factors or on dimensions. Distinction on factors is possible only for D-cases, since a distinction is a factor present in the precedent but not in NC, which helps the defendant. Where the distinction is "significant", we shall wish to emphasise this, using the algorithm ES given below.

Factor Distinction Algorithm (FD)

```
FD1 If case is AD, ABD, or ACD
      Write citation for cited case
      Write "is distinguishable because"
      For D-Factor in cited case
      If D-factor is a d-factor
        Write "in" NC
        Write D-factor
        Write "Not so in"
        Write cited case
        DO ES
      Else
        Write "In" cited case
        Write D-factor
        Write "Not so in"
        Write NC
        DO ES
```

If we have a factor distinction we can emphasise its significance by making use of the factor hierarchy. The Emphasise significance algorithm presented here – and the Downplay Significance algorithm described later – are simplifications of the algortihms implemented in the CATO program. The CATO algorithms take account of various strategic considerations to determine which abstract factor in terms of which to characterise the factor to be emphasised or downplayed. For details, see Aleven (1997). The simplification is made here to allow us to ignore heuristic aspects. Using the simplification, we need to find the highest abstract factor supported by the factor we wish to emphasise, which is not supported by any factor in NC. An abstract factor is a +parent of a supporting child factor and a –parent of a child factor which militates against it.

Emphasise significance algorithm (ES)

```
ES1 Set fac = factor to emphasise
ES2 Set best-parent = null
ES3 Find +parent of fac: if none return best-parent
ES4 If +parent is not a +parent of any factor in NC
        set best parent = +parent
        set fac to +parent
        GOTO ES3
      Else return best-parent
ES5 If best-parent not null
      If factor is a d-factor
        write "In" name of NC
        write factor
        write "this was not so in" name of cited case
        write "This is a marked distinction. It shows"
        write "that in" name of NC
        write best-parent
        write "This was not so in" name of cited case
ES6 Else
      write "In" name of cited case
      write factor
      write "this was not so in" name of NC
      write "This is a marked distinction. It shows"
      write "that in" name of cited case
      write best-parent
      write "This was not so in" name of NC
```

To distinguish on dimensions we must consider each dimension shared by NC and the cited case (A-dimension or B-dimension). Distinguishing is possible if the value for the dimension is more favourable to the plaintiff in the cited case than in NC.

```
Dimension Distinction Algorithm(DD)
    DD1 For each A-dimension
        If A-dimension more pro plaintiff in cited case than NC,
           Write "In" name of cited case
           Write A-dimension
           Write "was more favourable to the plaintiff"
    DD2 For each B-dimension
        If B-dimension more pro-defendant in NC than cited case
           Write "In the current case"
           Write B-dimension
           Write "is more favourable to the defendant"
```

This completes the response for the defendant.

In preparing the rebuttal, the advocate for the plaintiff will make use of moves C2, C3, C5 and C6.

```
Rebuttal Algorithm (RR)
    RR1 Distinguish counter examples
    RR2 Downplay distinctions
    RR3 Emphasise strengths
    RR4 Show weaknesses not fatal
```

C2 is used to rebut counter examples. If the counter example is a C-case with respect to NC, we can distinguish on those factors:

```
Factor Rebuttal Algorithm (FR)
   FR1 If counter example is AC, ABC, or ACD
       Write citation for counterexample
       Write "is distinguishable because"
       For C-Factor in counterexample
       If C-factor is a p-factor
          Write "In" NC
          Write C-factor
          Write "Not so in"
       Write name of counterexample
       Else Write "In" counterexample
            Write C-factor
            Write "Not so in"
            Write N
```

We can also distinguish using dimensions, if an A-dimension or a B-dimension is more favourable to the plaintiff.

```
Dimension Rebuttal Algorithm (DR)
    DR1 For each A-dimension
        If A-dimension more pro plaintiff in NC than counterexample,
           Write "In the current case"
           Write A-dimension
           Write "is more favourable to the plaintiff"
    DR2 For each B-dimension
```

```
If B-dimension more pro-defendant in counterexample than NC
   Write "In" name of counterexample
   Write B-dimension
   Write "was more favourable to the defendant"
```

To implement move C3, where we wish to downplay the significance of a distinction, we make use of the factor hierarchy in a manner similar to ES above. Here we need to find the lowest abstract factor which the factor to diminish supports and which is also supported by a factor in NC.

Downplay Significance algorithm

```
DS1 Set fac = factor to downplay
DS2 Set best-parent = null
DS3 Find +parent of fac: if none return best-parent
DS4 If +parent is a +parent of any factor, F, in NC,
          set best parent = +parent
          set fac to F
          return best-parent
    Else set fac to +parent
    GOTO DS3
DS5 If best-parent not null
    If factor is a p-factor
      write "In" name of cited case
      write factor
      write "this was not so in" name of NC
      write "However, this is not a significant distinction"
      write "In" name of NC
      write fac
      write "Therefore in both cases"
      write best-parent
DS6 Else
DS7   write "In" name of NC
      write factor
      write "this was not so in" counter-example
      write "However, this distinction is not significant"
      write "In" name of NC
      write fac
      write "therefore in both cases"
      write best-parent
      write "yet plaintiff may still win"
```

We now have the opportunity, assuming that we have some C-factors, to emphasise strengths. For each C-factor we need to find a case such that it is a citeable case, the C-factor is an A-factor in that case, if it is a p-factor, or a B-factor if it is a d-factor.

Up-play strengths algorithm (UA)

```
UA1 Do AA to produce CC-list
UA2 For each C-factor F, such that F is a p-factor
        Find best case, B, in CC-list such that F is an A-factor
UA3 Write "Court should find for the plaintiff where"
UA4 Write "In addition to"
UA5 Write A-factors in B other than F
UA6 Write "Also" F
UA7 If B-factors in NC and B
      Write "Even though"
      Write B-factors in NC and B
```

UA8 Cite B
UA9 For each C-factor C, such that C is a d-factor
    Find best-case, B, in CC-list such that C is a B-factor
UA10 Write "Court should find for the plaintiff where"
UA11 Write "In addition to"
UA12 Write A-factors in B
UA13 Write "also not" F
UA14 If B-factors in NC and B
    Write "Even though"
    Write B-factors in NC and B
UA15 Cite B

Finally we can attempt to show that any weaknesses represented by D-factors are not fatal by citing p-cases where these factors also applied. For a d-factor, we need a case for which this is a B-factor with respect to NC, and for a p-factor a case where this factor was also missing: which means a case for which this factor is also a D-factor with respect to the cited case. We should, if possible also point to ANY A-factors shared with NC:

Weaknesses not fatal algorithm (WNF)

WNF1 For p-factor, F, in D-factors for cited case with respect to NC
    Find case, C, where F is an D-factor for the cited case with respect to C
WNF2 Order on number of A factors in C with respect to NC
WNF3 Select best C
WNF4 Write "The absence of"
WNF5 Write F
WNF6 Write "Does not preclude a finding for the plaintiff
WNF7 If A-factor in C with respect to NC
    Write "especially where"
    Write shared A-factors
WNF8 Cite C.
WNF9 For d-factor, F, in D-factors for cited case with respect to NC
    Find case, C, where F is a B-factor with respect to NC
WNF10 Order on number of A factors in C with respect to NC
WNF11 Select best C
WNF12 Write F
WNF13 Write "does not preclude a finding for the plaintiff"
WNF14 If A-factor in C with respect to NC
    Write "especially where"
    Write shared A-factors
WNF15 Write "cite" C.

This completes the argument. See Figure 2 for a diagrammatic overview of the whole procedure.

7. An Example

For my example I shall use that provided in Prakken and Sartor (1997). It has the advantage of brevity, and is completely available, whereas the information used by HYPO/CATO is not. I have slightly adapted the example, with some changes of names so as to avoid the impression that pro-plaintiff factors are the negation of pro-defendant factors and to include a dimension.
The issue they explore is whether, in the context of some hypothetical tax law a person's domicile has been changed. We will assume that pro-change is pro-plaintiff. The factors they identify are:

Figure 2: Three-Ply Algorithm

f1 kept-house (d)
f2 gave-up-house (p)
f3 domestic-employer (d)
f4 duration-of-absence: dimension measured in months:
pro plaintiff direction is more days.
f5 company-has-domestic-property (d)
f6 company-has-foreign-property (p)
f7 company-has-domestic-headquarters (d)
f8 company-has-foreign-headquarters (p)
f9 company-has-foreign-president (p)
f10 no-domestic-job-prospects (p)
f11 domestic-job-prospects (d)
f12 foreign-car (p)

Of these f3 is an abstract factor, supported by f5 and f7 and militated against by f6, f8 and f9.

Prakken and Sartor give four precedents and a new case. Note that I shall replace the abstract factor f3, where it occurs in their case descriptions, by its constituent factors f5-f9. I shall also assign a specific number for duration (f4).

case	f1	f2	f4	f5	f6	f7	f8	f9	f10	f11	f12	outcome
P1	0	1	36	0	1	0	1	0	0	0	0	p
P2	1	0	?	1	0	0	1	1	0	0	0	p
P3	0	1	12	1	0	1	0	0	0	0	0	d
P4	0	1	7	0	0	0	0	0	1	0	1	p
NC	0	1	28	1	0	1	0	0	1	0	0	?

Suppose we wish to argue for change. We determine A-D factors for each case with respect to NC as follows:

case	A			B		C		D			
P1	f2	f4				f10		f5	f6	f7	f8
P2				f5		f1	f2		f7	f8	f9
P3	f2	f4		f5	f7	f10					
P4	f2	f4	f10					f5	f7	f12	

We have two p-cases that are also A-cases, P1 and P4. Both are also D-cases. P1 is, however, also a C-case, so we select P1 as our first case to cite.

We apply SP to get

> Court should find for the plaintiff. Where gave-up-house and duration-of-absence = 28. Cite P1.

We now attempt a response. Applying CE, P3 appears to be more on point. Applying FD we can distinguish on all of factors f5, f6, f7 f8. Applying ES we can emphasise these f5 an f7 by pointing to abstract factor f3. Applying DD we find that f4 is more favourable to the plaintiff in P1 than NC This yields the following response.

> Counter example: P3 is more on point an held for the defendant where it was also the case that company-has-domestic-property and company-has-domestic-headquarters.
>
> P1 is distinguishable because:
>
> In NC company-has-domestic-property. Not so in P1.
>
> This is a marked distinction: it shows that in NC domestic-employer. This was not so in P1.
>
> In NC company-has-domestic-headquarters. Not so in P1.
>
> This is a marked distinction: it shows that in NC domestic-employer. Not so in P1.
>
> In P1 company-has-foreign-property. Not so in NC.
>
> In P1 company-has-foreign-headquarters. Not so in NC.
>
> In P1 duration was more favourable to the plaintiff.

Finally we construct the rebuttal. Using FR we can distinguish the counter example on f10. Using DR we can distinguish on duration which is considerably more favourable to plaintiff in NC than P3. We cannot, however, downplay the distinction with regard to f5-f8. We can, however, cite P4 to emphasise the additional strength with respect to P1 represented by f10, and P2 to suggest that f5 is not fatal. We can also use P4 to show that the absence of f6 and f8 is not fatal.

> P3 is distinguishable because: In NC no-domestic-job-prospects. Not so in P3.
>
> In P3 duration was more favourable to the defendant.
>
> Court should find for plaintiff where in addition to gave up house and duration-of-absence = 28 also no-domestic-job-prospects. Cite P4.
>
> The absence of company-has-foreign-property and company-has-foreign-headquarters does not preclude a finding for the plaintiff especially where gave up house and duration-of-absence = 28 and no-domestic-job-prospects. Cite P4
>
> company-has-domestic-property does not preclude a finding for the plaintiff. Cite P2.

Who has won the argument? That is essentially a matter of judgement, depending on whether we consider the counter example sufficiently distinguished, and the additional strengths sufficiently persuasive. There is no compulsion to go one way or the other, but the argument clearly sets out the issues that must be resolved. Since the argument is inconclusive, we may wish to see how it would have gone if

we had started from P4, which was the source of a strong rebuttal in the previous argument.

> Court should hold for the plaintiff. Where gave-up-house and duration-of-absence = 28 and no-domestic-job-prospects. Cite P4.

The response this time has no counter examples since the only available d-case lacks f10. We can, however, distinguish the case:

> P4 is distinguishable because
>
> In NC company-has-domestic-property. Not so in P4.
>
> This is a marked distinction. It shows that in NC domestic-employer. This was not so in P4.
>
> In NC company-has-domestic-headquarters. Not so in P4.
>
> This is a marked distinction. It shows that in NC domestic-employer. This was not so in P4.
>
> In P4 foreign-car. Not so in NC.

Note here duration is more favourable for the plaintiff in NC than in P4 and so it cannot be used to distinguish the case.

In the rebuttal we have no counter example to distinguish and no strengths to emphasise, since f10 has already played its part in blocking the counter-example. We again cannot downplay the significance of f5 and f7. We can, however, suggest that f5 is not fatal:

> company-has-domestic-property does not preclude a finding for the plaintiff. Cite P2.

Note that in this argument P1 is not mentioned. The points for the plaintiff in P1 are subsumed in P4, and P1 contains additional D-factors which make it vulnerable to attack.

Again, however, the argument is inconclusive, and we must weigh its strengths. This should, not, however, be seen as a problem; interesting cases where CBR is appropriate typically will come down to a judgement call as to the competing strengths of the arguments pro and con. Which argument we choose is a tactical matter, in the first argument we start with a case offering quite basic support, allowing the defendant a strong response, which we hope to counter with the strong rebuttal of P4. In the second we display our best case first, and accept a weaker response to which there is a thinner rebuttal.

8. Conclusions and Future Work

The purpose of this paper is to give a clear statement of the current understanding of how to reason with cases in AI and Law. I believe that it demonstrates that we in fact understand quite a lot, and CBR systems can be readily implemented. An important point, however, is the heavy reliance placed on factors, which in turn involves an extensive analysis of the cases in the domain to identify the factors and to group them into a factor hierarchy. This in turn means that the domain must be susceptible to this kind of analysis. So far it has been undertaken in the areas of trade secrets (Ashley 1990, and Aleven and Ashley 1997), and in home office tax deductions, (Rissland et al 1996). Arguably some domains (e.g. much of social security) would not respond to this kind of analysis, turning rather on simple

matters of fact. If this is so, then such domains are probably not best served by CBR, but are better modelled using rule based techniques.

A secondary aim of the paper was to clarify that nature of reasoning with cases. Much of the work often considered under the heading of CBR is rather concerned with finding cases, and is better considered as information retrieval than CBR. Also I have tried to distinguish CBR from reasoning with information derived from cases. In CBR the integrity of the case matters, and the way cases are use depends as much on the context supplied by the case as by the leading principle that it seems to support. In the second argument above P1 is not cited not because it fails to yield a useful principle, but because other features of the case make it too vulnerable to attack. Reasoning with cases involves important rhetorical aspects as well as logical aspects, and these rely crucially on the fact that they are whole cases rather than another source of principles.

To conclude I wish to identify three areas of future work which I believe will be fruitful.

1. The algorithm could be implemented in a number of domains. In order to see whether the insights it offers are real we need much more experience of its application. It would also be interesting to see whether there are non-legal areas of dispute to which it could be applied.
2. We can see an argument as a style of document. In a very interesting paper (Branting et al 1997) a proposal for document generation is made which separates out the illocutionary structure of a document from its rhetorical structure. If we take the argument skeleton provided by the algorithm as the rhetorical structure and the various argument moves as providing the illocutionary structure, these ideas could be applied in this area, and perhaps give a better handle on the interplay between logic and rhetoric.
3. Finally we have used the eight arguments of CATO. Other argument moves have been identified in Skalak and Rissland (1992) and Rissland et al (1996), and it would be most interesting to investigate whether these moves can be subsumed under the eight in CATO, or, failing that whether they could be incorporated into the general framework as extensions to the algorithm.

Argument, and argument from cases in particular, is central to AI and Law. Significant progress has been made, and it is important that this progress is consolidated and advanced.

References

Aleven, V., 1997, Teaching Case-Based Argumentation Through a Model and Examples PhD Dissertation, University of Pittsburgh Graduate Program in Intelligent Systems.

Aleven, V., and Ashley, K.D., 1997, Evaluating a Learning Environment for Case-Based Argumentation Skills, in Proceedings of the Sixth International Conference on Artificial Intelligence and Law, Melbourne, ACM Press, New York. pp 170-179.

Ashley, K.D., 1990, Modeling Legal Argument MIT Press, Cambridge.

Branting, L.K., Lester, J.C., and Callaway, C.B., 1997 Automated Drafting of Self-Explaining Documents in Proceedings of the Sixth International Conference on Artificial Intelligence and Law, Melbourne, ACM Press, New York. pp 72-81.

Bruninghaus, S., and Ashley, K.D., 1997, Finding Factors; Learning to Classify Case Opinions Under Abstract Fact Categories, in Proceedings of the Sixth International Conference on Artificial Intelligence and Law, Melbourne, ACM Press, New York. pp 123-131.

Kowalski, A., 1991, Case-based Reasoning and the Deep Structure Approach to Knowledge Representation in Proceedings of the Third International Conference on AI and Law, ACM Press: New York.

Montazeri, M.A., Bench-Capon, T.J.m., and Adam, A.E., 1997, LASER: A System to Retrieve UK Employment Law Cases, Information and Communications Technology Law, Vol 6 No 1. pp 41-54.

Prakken, H., and Sartor, G., 1996, A System for Defeasible Argumentation, With Defeasible Priorities, in Proceedings of the International Conference on Formal and Applied Practical Reasoning, Springer Lecture Notes in AI 1085, Springer Verlag, Bonn. pp 510-524.

Prakken, H., and Sartor, G., 1997, Reasoning with Precedents in a Dialogue Game, in Proceedings of the Sixth International Conference on Artificial Intelligence and Law, Melbourne, ACM Press, New York. pp 1-9.

Rissland, E.L., Skalak, D.B., and Friedman, M. Timur, 1996, BankXX: Supporting Legal Arguments Through Heuristic Retrieval, Artificial Intelligence and Law, Vol 4 No 1. pp 1-71.

Skalak, D.B., and Rissland, E.L., 1992, Arguments and Cases: An Inevitable Intertwining, Artificial Intelligence and Law, Vol 1 No 1 pp 3-42.

Stranieri, A., and Zeleznikow, J., 1995, The Split-Up system: Integrating neural networks and rule- based reasoning in the legal domain in Proceedings of the Fifth International Conference on AI and Law, ACM Press: New York.