**Artificial Intelligence and Law manuscript No.**
(will be inserted by the editor)

# A Methodology for Designing Systems to Reason with Legal Cases Using Abstract Dialectical Frameworks

**Latifa Al-Abdulkarim · Katie Atkinson · Trevor Bench-Capon**

September 30, 2016

**Abstract** This paper presents a methodology to design and implement programs intended to decide cases, described as sets of factors, according to a theory of a particular domain based on a set of precedent cases relating to that domain. We use Abstract Dialectical Frameworks (ADFs), a recent development in AI knowledge representation, as the central feature of our design method. ADFs will play a role akin to that played by Entity-Relationship models in the design of database systems. First, we explain how the factor hierarchy of the well-known legal reasoning system CATO can be used to instantiate an ADF for the domain of US Trade Secrets. This is intended to demonstrate the suitability of ADFs for expressing the design of legal cased based systems. The method is then applied to two other legal domains often used in the literature of AI and Law. In each domain, the design is provided by the domain analyst expressing the cases in terms of factors organised into an ADF from which an executable program can be implemented in a straightforward way by taking advantage of the closeness of the acceptance conditions of the ADF to components of an executable program. We evaluate the ease of implementation, the performance and efficacy of the resulting program, ease of refinement of the program and the transparency of the reasoning. This evaluation suggests ways in which factor based systems, which are limited by taking as their starting point the representation of cases as sets of factors and so abstracting away the particular facts, can be extended to address open issues in AI and Law by incorporating the case facts to improve the decision, and by considering justification and reasoning using portion of precedents.

**Keywords** Legal Reasoning, Factors, ADF, Knowledge Engineering, Argumentation

## 1 Introduction

Modelling reasoning with legal cases has been the central question for many researchers in AI and Law. Early work such as [55] and [42] identified important research issues, and since

Latifa Al-Abdulkarim
Department of Computer Science. University of Liverpool. E-mail: latifak@liverpool.ac.uk

Katie Atkinson
Department of Computer Science. University of Liverpool. E-mail: katie@liverpool.ac.uk

Trevor Bench-Capon
Department of Computer Science. University of Liverpool. E-mail: tbc@liverpool.ac.uk

then several developments have established a good degree of consensus as to an effective model of legal reasoning with cases. First, facts must be determined on the basis of evidence (e.g. [25]). Next these facts must be used to ascribe legally significant predicates to the case. These predicates, which serve as intermediaries between the world of fact and the world of law (e.g. [30]), have been termed intermediate concepts (e.g.[54], [10]), but are more often called *factors*, following the highly influential systems HYPO [9] and CATO [5], which are still foundational for discussion of reasoning with legal cases in AI and Law. These factors are stereotypical patterns of fact, present or absent in a case, which favour one or other of the parties to the case. In CATO these factors were formed into a hierarchy in which the *base-level factors* were the children of *abstract factors*, the presence of a child serving as a reason for or against the presence of the parent (depending on the party favoured by the two factors). The factors are then used to establish the legal conclusions (see e.g. [34], [54], [47]).

Further work has applied formal argumentation models to legal argumentation. For example, the abstract argumentation framework of Dung [40] has been applied to law in [58] and many subsequent papers, and an extension to accommodate values [12] was applied to legal argument in [15] and elsewhere. The computational and theoretical strands of factor based reasoning have, however, by no means remained separate, and there have been a number of attempts to express the factor based reasoning embodied in systems such as CATO in a rule based format suitable for representation in a more formal framework (e.g. [60]), [48]).

In this paper, we propose a new approach to the design and implementation of systems to reason with legal cases using a powerful generalisation of Dung's abstract argumentation frameworks [40], *Abstract Dialectical Frameworks* (ADF) [33] [32][1]. ADFs generalise the abstract argumentation frameworks introduced by Dung by replacing Dung's single acceptance condition (that all attackers be defeated) with acceptance conditions local to each particular node. The nodes in an ADF are statements, not necessarily arguments, and can be related by a variety of types of links, not just attackers. We use the ADFs as the central design element of our systems, able to mediate between the case analysis and the executable program, and to encompass both frame based and rule based styles of system.

From the ADF definition, we can see its similarity to the structure of the factor hierarchy in CATO (as shown in figures 1 and 2). Moreover, the additional flexibility that ADFs give over AFs, which were used to represent a body of precedent cases in, e.g. [16] and [17], allows a more natural representation than has been possible in AFs, especially since we have both pro and con reasons. Therefore, in this paper we will design our systems by instantiating a factor hierarchy for the domain as an ADF in order to encapsulate the theory of case law for the legal domain that we wish our program to apply to. Recognising that the case law of a legal domain goes through a life cycle of three stages as stated by Levi [53], we intend this approach to be applied to the second stage where there is a period of stability and the theory is settled. As with legislation [29], it is only when the law is a settled state that the investment in implementing a program becomes justifiable. The methodology described in this paper goes through three iterated phases:

**Domain analysis and representation:** The approach is applied to three domains. The first starts with the use of the existing knowledge of US Trade Secrets in CATO and IBP. For a second domain, we construct ADFs in the domain of Wild Animal cases (*Popov v Hayashi* and related cases as modelled in [18]). Finally, we use a set of cases from the Automobile Exception to The Fourth Amendment which has been discussed in the literature in [62], [43], [13] and [3]. The representation in those papers is, however,

[1] ADFs are formally defined in Definition 1, section 2.3.

mostly to illustrate particular points and hence is incomplete, and does not descend into the fine details. We provide our own new, complete, fine grained, analysis, which we use as the design for our implementation.

**Implementation and Refinement:** Next, we move from the analysis to an executable program in a direct and immediate way using Prolog. We translate the ADF acceptance conditions into Prolog procedures, by expressing the conditions in Prolog syntax and adding some standard control and reporting information to each clause, and then execute the resulting program. Comparison of the program results with the actual outcomes of the cases allows the theory to be refined, by returning to the original decision texts and reconsidering the analysis, exploiting the software engineering benefits afforded by the ADF representation.

**Evaluation:** For each domain, we evaluate the ease of implementation, the performance and efficacy of the resulting program, the ease of refinement of the program and the transparency of the reasoning. The evaluation suggests ways in which better systems could be developed by incorporating the case facts which justify the base-level factors ascribed to the cases, and by justifying the reasoning using portions of precedents [27].

The outline for this paper is as follows. First we provide some background, comprising a discussion of system design aspects of engineering AI and Law systems, recapitulation of factor based systems for reasoning with legal cases, especially CATO and IBP and their factor hierarchies, and an overview of ADFs. After that, we present an ADF encapsulating the factor hierarchy of CATO showing the natural mapping of the statements and links from the factor hierarchy to the ADF and explaining the approach we follow in defining the acceptance conditions. Next, we apply the approach to two more legal domains showing the analysis and the representation of each domain, examples from the programs and their output, and the results and the possible refinements required to improve the results. Finally we evaluate the approach, discuss some further lessons learned from the exercise and conclude with suggestions for future work.

## 2 Background

In this section we will discuss the design of AI and Law systems, recapitulate the factor based hierarchies of CATO [5] and IBP [34] and provide the essentials of ADFs [32].

### 2.1 Design of AI and Law Systems

In the middle eighties, it was believed that the use of declarative representations would allow a straightforward implementation of AI and Law programs. For example, [64] argued that legislation could be directly implemented using Prolog, and then executed as an expert system to support decision making under the Act so represented. While this approach seemed to work for very small pieces of simply drafted legislation such as the British Nationality Act, capable of being implemented by a single person in a relatively short time, the approach did not scale to large, more complex, pieces of legislation, requiring a team of developers. For example, an attempt to apply the direct encoding approach to the Supplementary Benefits Act 1980 [22], which represented a much larger, much amended, more complex piece of legislation, the modelling of which took a team of two people several months, showed that without some design principles and the ability to communicate the work at a level above

code, allowing the work to be sensibly divided and co-ordinated, results were not satisfactory. In response to this, principles such as hierarchical formaisation [63] and isomorphism [19] were developed.

These guiding principles were still not really sufficient to provide adequate support for the design of substantial systems, capable of being shared, verified and reused, and providing a description of the program to serve as an entry point for those required to perform perfective and adaptive maintenance. As noted in [29], it is, given the liability of law to change over time, foolhardy to build a knowledge based system in Law unless there is a clear maintenance strategy. To provide support the notion of *ontology* [44] was adopted from general AI. Early examples were [67] and [52]. Ontologies were incorporated in AI and Law development methodologies (e.g. [66]), and remain an essential tool for the development of AI and Law systems today: workshops and tutorials on ontologies have been a regular feature of ICAIL conferences since 1995. Building a substantial knowledge based system in AI and Law without an ontology should now be seen as quite as misguided as trying to construct a database system without a Data Dictionary. Another approach to using software design tools to develop AI and Law systems was the Power project [65], which adapted UML (Universal Modelling Language) tools to model Dutch tax law in the development of AI systems. Without a well defined methodology, such a system could not have been attempted (and would not have been permitted) in the environment of a professional IT Department of a Government organisation, and is unlikely to have been as successful as the Power project proved to be. Another example of a well defined methodology is the approach used by Softlaw and its successor companies [50], [51] based on the *verbatim* approach proposed in [49].

All of the above referred to rule based systems, based on formalisations of legislation of some kind. The use of design tools to support reasoning with legal cases is much less common. Although there are ontologies for case based reasoning, such as [68], these have not been produced as part of the development of a substantial program development project. Also while ontologies are a useful design tool, they are not suitable for the design of all aspects of a system. Database design now uses several tools as standard (e.g. [39]), playing distinct but complementary roles, including *Data Dictionaries* and *Entity-Relationship Models*. As noted above, ontologies can be seen as playing a role similar of Data Dictionaries: here we propose that ADFs can provide design functions for knowledge based systems akin to those provided for database systems by Entity-Relationship Models.

Whereas ontologies capture the static relationships between objects in the domain through the definition of class-subclass and is-a relationships, as will be seen in section 2.3, ADFs can be used to capture the more dynamic aspects, including the inferential relationships. The children of ADF nodes are not subclasses, but the elements needed to decide the acceptability of the statement represented by the parent, and provide acceptability conditions for the parent in terms of these children. Thus while ontologies provide the vocabulary from which the rules can be constructed, ADFs determine the rules that will be required in the program and their relationships.

As such ADFs can both drive and record the design of the knowledge base for a system to apply a body a case law. In particular the otherwise monolithic rule base is modularised by being distributed as the acceptance conditions of nodes. Good modularisation - tightly coherent and loosely coupled - is an essential feature of good program design (see e.g. [61]).The modularisation of the knowledge base achieved by using an ADF is indeed tightly coherent since each set of rules is concerned only with the acceptance of a single statement, and contains all the rules needed to decide the acceptance of that statement. Loose coupling is achieved by limiting the components required to determine the acceptability of a node to the children of that node.

Knowledge Engineering advantages of ADFs include, in addition to the effective modularisation of the system:

- Effective partitioning of the problem space which limits the number of precedents required to determine the outcome of cases, as discussed in [2].
- Ready visualisation of the possible paths through a program: the connection between sets of rules is readily visible from the structure, whereas tracing which rules are invoked when a given rule is executed from a monolitic rule base of any reasonable size can often be difficult and error prone.
- Assurance of completeness, since it can readily been seen that each non-leaf node has its acceptance condition.
- Straightforward inclusion of additional nodes. Addition of nodes can be performed without fear that there will be unanticipated ramifications in the knowledge base.
- Awareness of what nodes will be affected by the removal (or modification) of a node.
- Support for verification: each node can be considered, either by reference to precedents or otherwise, in isolation, on its own merits.
- Neutrality and integration between frame based and rule representations. The node structure provides the former and the acceptance conditions the latter.
- Division of the labour across a team of implementors: realisation of the part of the program which decides one node can be done independently of the realisation of other nodes: how the fragments will link together is explicitly specified. The children determine the inputs required by each module (node): that is, how the modules interface with each other.

All of these things are highly desirable when designing and developing a system of any real size and substance to reason with legal cases. The lack of such support has been a significant barrier to the take-up of such systems in practice (see, e.g., [26] for the difficulties in developing a sizable knowledge base to represent case law without effective design support). In the detailed examples worked through in section 4, we believe that the efficacy of ADF with respects to these various aspects will become plain.

## 2.2 Factor Based Hierarchy Systems

We now describe CATO [5] and IBP [34], two of the best known and most substantial implementations of reasoning with legal cases. The anlysis reported in these papers will provide the basis for the design of our first system in section 4, and there is a thorough empirical evaluation of both systems in [34], which provides a very useful basis for comparison of system performance.

### *2.2.1 CATO*

CATO [5], which was a development from Rissland and Ashley's HYPO, most fully described in [9], takes as its domain US Trade Secret Law. CATO was primarily directed at law school students, and was intended to help them form better case-based arguments, in particular to improve their skills in distinguishing cases, and emphasising and downplaying distinctions. A core idea was to describe cases in terms of factors, legally significant abstractions of patterns of facts found in the cases, and to build these *base-level factors* into an hierarchy of increasing abstraction, moving upwards through intermediate concerns (abstract factors) to issues. An extract from the factor hierarchy, showing details of the support

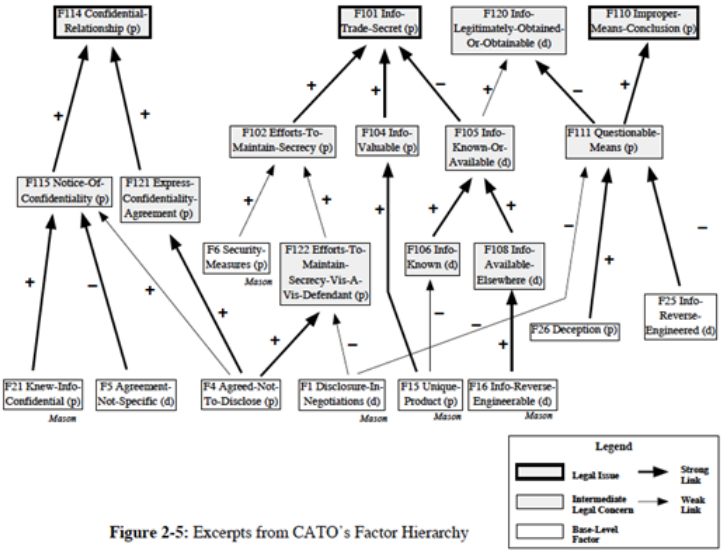


**Figure 2-5:** Excerpts from CATO's Factor Hierarchy

**Fig. 1** CATO Abstract Factor Hierarchy from [5]

and attack relations between the factors, is shown in Figure 1 and the complete hierarchy is shown in Figure 2. The Figures have been reproduced directly from [5]. Each factor favours either the plaintiff or the defendant. Like HYPO, the CATO program matches precedent cases with a current case to produce arguments in three plies: first a precedent with factors in common with the case under consideration is cited, suggesting a finding for one side. Then the other side cites precedents with factors in common with the current case but a decision for the other side as counter examples, and distinguishes the cited precedent by pointing to factors not shared by the precedent and current case. Finally the original side rebuts by downplaying distinctions, citing cases to prove that weaknesses are not fatal and distinguishing counter examples. CATO used twenty-six base level factors (there is no F9), as shown in Table 1. There is, however, no single root for the factor hierarchy as presented in [5]: rather we have a collection of hierarchies, each relating to a specific issue. To tie them together we turn to the Issue Based Prediction (IBP) system of Bruninghaus and Ashley [34], described in the next subsection.

### *2.2.2 Issue Based Prediction*

In IBP, which is firmly based on CATO, the aim is not simply to discover and present arguments, but to predict the outcomes of cases. To enable this, the issues of CATO's hierarchy are tied together using a logical model derived from the *Uniform Trade Secret Act*, which has been adopted by the majority of States in the US, and the *Restatement of Torts*. The model is shown in Figure 3. IBP used 186 cases in its very thorough empirical evaluation, 148 cases analysed for CATO and 38 analysed specifically for IBP. Unfortunately, the analyses of these cases are not all publicly available and so we will use the 32 cases harvested from a variety of public sources describing CATO and IBP by Alison Chorley and used to evaluate her AGATHA system [37], [36]. As part of the evaluation in [34], nine other sys-

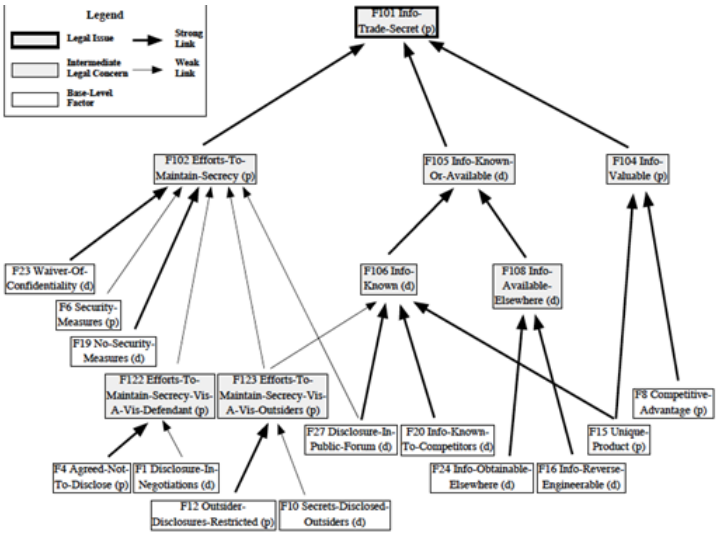


**Figure 3-2:** CATO's Factor Hierarchy, part 1

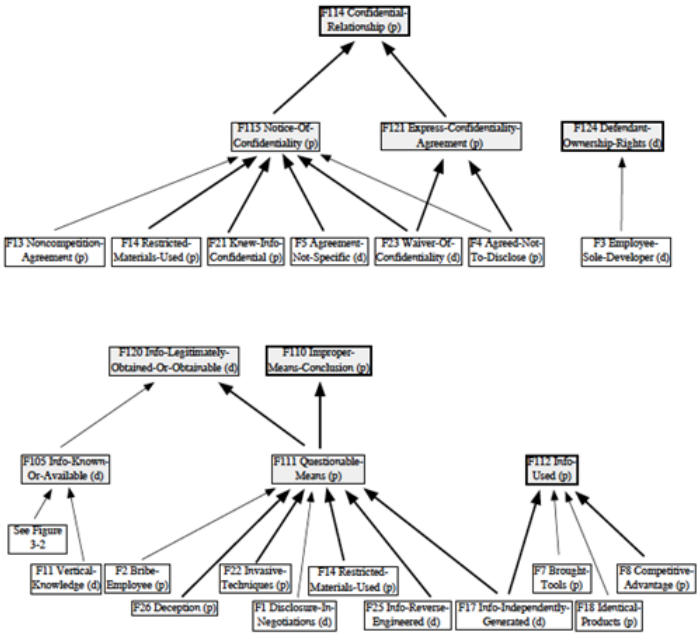


**Figure 3-3:** CATO's Factor Hierarchy, part 2

**Fig. 2** CATO Abstract Factor Hierarchy from [5]

| ID | Factor |
|---|---|
| F1 | DisclosureInNegotiations (d) |
| F2 | BribeEmployee (p) |
| F3 | EmployeeSoleDeveloper (d) |
| F4 | AgreedNotToDisclose (p) |
| F5 | AgreementNotSpecific (d) |
| F6 | SecurityMeasures (p) |
| F7 | BroughtTools (p) |
| F8 | CompetitiveAdvantage (p) |
| F10 | SecretsDisclosedOutsiders (d) |
| F11 | VerticalKnowledge (d) |
| F12 | OutsiderDisclosuresRestricted (p) |
| F13 | NoncompetitionAgreement (p) |
| F14 | RestrictedMaterialsUsed (p) |
| F15 | UniqueProduct (p) |
| F16 | InfoReverseEngineerable (d) |
| F17 | InfoIndependentlyGenerated (d) |
| F18 | IdenticalProducts (p) |
| F19 | NoSecurityMeasures (d) |
| F20 | InfoKnownToCompetitors (d) |
| F21 | KnewInfoConfidential (p) |
| F22 | InvasiveTechniques (p) |
| F23 | WaiverOfConfidentiality (d) |
| F24 | InfoObtainableElsewhere (d) |
| F25 | InfoReverseEngineered (d) |
| F26 | Deception (p) |
| F27 | DisclosureInPublicForum (d) |

**Table 1 Base Level Factors in CATO**

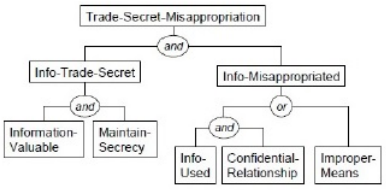


**Fig. 3** IBP Logical Model from [34]

tems were also considered to provide a comparison. Most of these were different forms of machine learning system, but programs representing CATO and HYPO were also included. IBP was the best performer: results reported in [34] for IBP, Naive Bayes (the best performer of the ML systems), CATO, HYPO and a version of IBP which uses only the model, with no CBR component, are shown in Table 2[2]. Direct comparison with AGATHA is hampered by the fact that evaluation in AGATHA was directed towards evaluating a variety of different heuristics and search algorithms used in that system, and so no version can be considered "definitive", and, of course, many fewer cases were used in the experiments. However, typically 27-30 of the 32 ($\approx 84-93\%$) cases were correctly decided by the theories produced by AGATHA [36].

[2] No explanation for using a different number of cases for CATO and IBP-model is given in [34].

| | correct | error | abstain | accuracy |
|---|---|---|---|---|
| IBP | 170 | 15 | 1 | 91.4 |
| Naive Bayes | 161 | 25 | 0 | 86.5 |
| CATO | 152 | 19 | 22 | 77.8 |
| HYPO | 127 | 9 | 50 | 68.3 |
| IBP-model | 99 | 15 | 38 | 72.6 |

**Table 2** Results from [34]

### 2.3 Abstract Dialectical Frameworks

Abstract Dialectical Frameworks (ADFs) were introduced in [33] and revisited in [32]. ADFs provide a generalisation of Dung's abstract argumentation frameworks (AFs) [40]. ADFs, like AFs, consist of a set of nodes and directed links between them, but whereas the links in an AF have a uniform interpretation, namely defeat, the links in an ADF can be given a variety of interpretations. Moreover, in ADFs the nodes are statements in general, rather than specifically abstract arguments. ADFs also generalise bipolar argumentation frameworks [35] which introduce a second relation (support) in addition to the attack relation of AFs. However, the ADF goes further and allows any type of link to be used. Although the acceptance conditions are often expressed as propositional functions, this need not be the case: all that is required is the specification of conditions for the acceptance or rejection of a node in terms of the acceptance or rejection of its children. In principle a "black box" program with the node as output and the children as inputs could be used, although this of course would not be suitable when using ADFs as a design aid.

ADFs are defined in [32] as follows:

**Definition 1**: An ADF is a tuple $ADF = < S, L, C >$ where
- $S$ is the set of statements (positions, nodes).
- $L$ is a subset of $S \times S$, a set of links.
- $C = \{C_{s \in S}\}$ is a set of total functions $C_s : 2^{par(s)} \rightarrow \{t, f\}$, one for each statement $s$. $C_s$ is called the acceptance condition of $s$.

In a Prioritised ADF, $L$ is partitioned into $L+$ and $L-$, supporting and attacking links, respectively.

**Definition 2**: A prioritized ADF (PADF) is a tuple $A = (S, L+, L-, >)$ where
- $S$ is the set of statements (positions, nodes).
- $L+$ and $L-$ are subsets of $S \times S$ the supporting and attacking links.
- $>$ is a strict partial order (irreflexive, transitive, antisymmetric) on $S$ representing preferences among the nodes.

We will use the partitioning into supporting and attacking links, but continue to see the acceptance conditions as specifying preferences locally rather than adopting a global ordering on nodes[3].

[3] Whereas PADFs were designed specifically to reflect Preference-Based Frameworks [6], the approach taken here reflects Value-Based Frameworks [12], which are more commonly used in AI and Law. The relationship between Preference- and Value-Based Frameworks is formally characterised in [56].

## 3 ADFs For Factor Hierarchy

These two ideas, the factor hierarchies of CATO and ADFs, can be brought together. Consider the CATO and IBP factor hierarchies, as shown in Figures 1, 2 and 3. We can now instantiate an ADF to encapsulate the legal domain knowledge as represented by these factor hierarchies as follows:

**Statement** The statements of the ADF form $S$, the set of all the issues, intermediate concerns and base level factors in the factor hierarchy.

**Links** is a subset of $S \times S$, a set of links where $L+$ are the supporting links labeled "$+$" and $L-$ are the attacking links labeled "$-$" in the figures.

**Acceptance Conditions** For each abstract factor (non-leaf node), we define acceptance conditions in terms of their supporting and attacking children intended to reflect the decisions in precedent cases. Often, however, the nature of the relationship is clear and so recourse to particular precedents is unnecessary. That precedents would not always be required to resolve the comparisons was recognised in [5]:

> "for certain conflicts, it is self evident how they should be resolved. ... It is not necessary to look to past cases to support that point." (p47).

The acceptance conditions expressed in this way form a set of tests. The order of the tests expresses preferences. If none are satisfied, the node (abstract factor) is assigned to a default value.

Although [33], [31] and [32] have shown ADFs to be formally sound and their formal properties have been investigated, we make little explicit use of these formal properties in this paper. Our ADFs are rather simple, since for the domains we consider our designers will produce cycle free ADFs. As such the grounded, preferred and stable semantics coincide. Moreover since our input cases will determine the labelling of base level factors (the leaf nodes), the nodes of our ADFs can be completely and unambiguously labelled using an efficient algorithm to find the grounded extension. This is important since it means that programs based on a design in the form of a cycle-free ADF are themselves tractably computable. In future work we will be looking at using implemented tools for processing ADFs to compare results under different semantics.

## 4 Applications

In this section we apply the approach to several different domains which have often been used in AI and Law. First we consider US Trade Secrets, using the 32 cases from [37]. Next, we extend the work to further domains: starting with a small scale domain concerning Wild Animals (5 cases) before proceeding to the Automobile Exception to The Fourth Amendment (10 cases). Note that our aim is only to encapsulate (rather than learn) a theory of the applicable law. For this a limited number of cases will suffice (cf. HYPO, which used fewer than 25 cases [9], and the discussion in [2] of how many precedents are needed).

The program is implemented using Prolog, taking advantage of its closeness to our expression of the acceptance conditions (effectively each test in an acceptance condition corresponds to a clause in the program), which made the implementation quick, easy, transparent and readily validated with respect to the design. The Prolog program was formed by ascending the ADF, rewriting the acceptance conditions as groups of Prolog clauses to determine the acceptability of each node in terms of its children. This required restating the tests using the appropriate syntax, adding some reporting to indicate whether the node is

satisfied (defaults are indicated by the use of "accepted that"), and some control to call the procedure to determine the next node, and to maintain a list of factors accepted so far. Some examples of the resulting code are given in section 4.1.2.

### 4.1 US Trade Secrets Domain

Using the complete factor hierarchy of [5] given in Figure 2, we can produce an ADF which has as its leaf nodes the base level factors of CATO. This is described in tabular form in Table 3. The roots of CATO's hierarchies correspond to the leaves of the IBP logical model: we can therefore combine them into a single ADF by using this structure. The relevant additions to the ADF needed to integrate the IBP model are shown in Table 4 (note that F124 is not discussed in [34]).

| ID | *S* | *L*+ | *L*− |
|---|---|---|---|
| F102 | EffortstoMaintainSecrecy | F6, F122, F123 | F19, F23, F27 |
| F104 | InfoValuable | F8, F15 | F105 |
| F105 | InfoKnownOrAvailable | F106, F108 | |
| F106 | InfoKnown | F20, F27 | F15, F123 |
| F108 | InfoAvailableElsewhere | F16, F24 | |
| F110 | ImproperMeans | F111 | F120 |
| F111 | QuestionableMeans | F2, F14, F22, F26 | F1, F17, F25 |
| F112 | InfoUsed | F7, F8, F18 | F17 |
| F114 | ConfidentialRelationship | F115, F121 | |
| F115 | NoticeOfConfidentiality | F4, F13, F14, F21 | F5, F23 |
| F120 | LegitimatelyObtainable | F105 | F111 |
| F121 | ConfidentialityAgreement | F4 | F23 |
| F122 | MaintainSecrecyDefendant | F4 | F1 |
| F123 | MaintainSecrecyOutsiders | F12 | F10 |
| F124 | DefendantOwnershipRights | F3 | |

**Table 3 CATO as ADF**

| ID | *S* | *L*+ | *L*− |
|---|---|---|---|
| F200 | TradeSecretMisappropriation | F201, F203 | F124 |
| F201 | InfoMisappropriated | F110, F112, F114 | |
| F203 | InfoTradeSecret | F102, F104 | |

**Table 4 IBP Logical Model as an ADF**

In Tables 3 and 4 we can see that we have eighteen nodes to provide with acceptance conditions. One (F124) has only a single supporting child: thus the acceptance condition will be $Parent \longleftrightarrow Child$. We will write this (and the other acceptance conditions) as a set of tests for acceptance and rejection, to be applied in the order given, which allows us to express priority between them. We choose this form of expression because we find it easier to read in many cases, because it corresponds directly to the defeasible rules with priorities used in formalisms such as ASPIC+ [59], and because it directly corresponds as Prolog code. The last test will always be a default. Thus we write $Parent \longleftrightarrow Child$ as

```
Accept Parent if Child.
Reject Parent.
```

Where NOT is required we use negation as failure. The tests are individually sufficient and collectively necessary, ensuring equivalence with the logical expression (see [38]). Six nodes (F201, F203, F105, F108, F114 and F124) have only supporting links: these can be straightforwardly represented using AND and OR. We followed the IBP model for the two nodes taken from that model (F201 and F203), and used OR for the other four. The most complicated was InfoMisappropriated (F201):

```
Accept InfoMisappropriated if F114 AND F112.
Accept InfoMisappropriated if F110.
Reject InfoMisappropriated.
```

Five nodes have one supporting and one attacking link. These are best seen as forming an exception structure: accept (reject) the parent if and only if supporting (attacking) child unless attacking (supporting) child. Note that the exception may be the supporting *or* the attacking child: in the former case the default will be *reject*, and in the latter the default will be *accept*. Thus:

```
Accept Parent if Supporter AND (NOT Attacker).
Reject Parent.
```

or

```
Reject Parent if Attacker AND (NOT Supporter).
Accept Parent.
```

For F110, F120 and F121 the attacking child is the exception, while for F122 and F123 the supporting children are the exceptions. This leaves seven nodes. For F200 we regard the attacking link as an exception to the case where the conjunction of the supporting links holds:

```
Accept Trade Secret Misappropriation if
                   Info Trade Secret AND
                   Info Misappropriated AND
                   (NOT Defendant Ownership Rights).
Reject Trade Secret Misappropriation.
```

For F104 and F112 we see the supporting links as offering disjoint ways of accepting the parent, and the attacking child as a way of establishing that the factor is not present. We default to *yes* because in many of the cases there are no factors for either side present relating to this point, and yet the abstract factor is required to be present. We take it that this factor was often simply accepted on the facts and uncontested, and so there was no discussion on the point, and so no factors were explicitly mentioned in the decision. Thus for F112:

```
Accept F112 if F18.
Accept F112 if F8.
Accept F112 if F7
Reject F112 if F17.
Accept F112.
```

The remaining four are more complicated because they involve more factors. This might be where we should use some reasoning with precedents, but here (because we have only a very limited number of precedents) our approach is to make an initial attempt to supply tests, and remain prepared to adjust these in the light of particular precedents where necessary. For F106 (*InfoKnown*) we use

```
Accept F106 if F20.
Accept F106 if F27 AND (NOT F15).
Accept F106 if F27 AND (NOT F123).
Reject F106.
```

The rationale is that if the information is known to competitors, it is known, but even if it is disclosed in a public forum, the uniqueness of the product can suggest that the disclosure had no impact (i.e. it was not sufficiently widely known), and so the secret remained effectively unknown (and so F24 (InformationObtainableElsewhere) is more appropriate). The third clause suggests that the public disclosure might be restricted (e.g. if the secret was disclosed in a court of law during a trade secrets hearing), so that the information may be known, but embargoed.

For F115, we regard each of the four supporting links (F4, F13, F14 and F21) as distinct ways of establishing notice of confidentiality. F23 (WaiverOfConfidentiality) is an exception to all of them whereas F5 (AgreementNotSpecific) is an exception only to F4 and F13, since the other two do not relate to agreements.

```
Reject F115 if F23.
Accept F115 if F21.
Accept F115 if F14.
Reject F115 if F5.
Accept F115 if F4.
Accept F115 if F13.
Reject F115.
```

Similarly for F111 (QuestionableMeans) we see the supporting links as four different ways in which questionable means can be established. The attacking links here seem like counter claims rather than exceptions, and suggest that for this node we may eventually need to explore precedents to identify preferences. However, as a first attempt we will regard them as three distinct ways of rejecting the claim. We thus have seven clauses, one for each factor, which we initially order as F25, F17, F22, F26, F14, F2, F1, to reflect the strong and weak links shown in CATO.

```
Reject F111 if F25.
Reject F111 if F17.
Accept F111 if F22.
Accept F111 if F26.
Accept F111 if F14.
Accept F111 if F2.
Reject F111 if F1.
Reject F111.
```

This leaves F102, EffortsToMaintainSecrecy, which has three supporting and three attacking links. F19 is applicable only if no security measures at all are taken, which suggests that it has priority. Similarly a waiver of confidentiality (F23) or disclosure in a public forum (F27) could be seen as negating any efforts to maintain secrecy, although F123 provides a possible exception to the latter. The remaining two supporting links we also regard as independent. Thus here we have six clauses, offering reasons to reject or accept, ordered F19, F23 (F27 and NOT F123), F6, F122, F123 with *reject* as the default.

```
Reject F102 if F19.
Reject F102 if F23.
```

```
Reject F102 if F27 and NOT(F123).
Accept F102 if F6.
Accept F102 if F122.
Accept F102 if F123.
Reject F102.
```

#### *4.1.1 Relation to Structured Argumentation*

We have realised our tests in a Prolog program, to provide an executable, testable, instantiation of the analysis. The overall output of our system bears a strong resemblance to the kind of structured argumentation found in ASPIC+ [59]. The union of the acceptance conditions can be seen as the knowledge base required by the ASPIC+ framework. Determining the acceptance or rejection of the various nodes produces sub-arguments, and these can be linked to produce the argument for finding for the plaintiff (or defendant) which follows the argument-subargument structure of ASPIC+ arguments. There are also differences: because the ordering of clauses expresses priority between arguments, only the winning arguments are generated. Thus the output does not include all arguments, but only the winning line of argument. Where, however, potential attackers are children rather than siblings, but are not acceptable, this is reported. Thus although there are correspondences, especially through the argument-subargument structure, the control regime employed by the program means that there are also important differences. These relate mainly to conflicts: the output shows only the winning side of the case, and does not provide a good record of the rejected arguments available to the losing side. This is because our program is only intended to produce an outcome for the case, (as with IBP) rather than a set of arguments (as with HYPO). None the less, the correspondences are such that this is worth exploring further in future work, using a different control regime in the implementation to force the generation of all arguments, as a potentially fruitful way of formalising the entire reasoning. This, however, will be a different program, one directed towards the HYPO task. Henceforth we will concentrate on the actual realisation of the decision making program in Prolog.

#### *4.1.2 Prolog Program*

In this section we will give some examples of the code. The program is executed by posing a query which gets the first factor to be decided with the case and its base-level factors as arguments. Each factor is called in turn, with the factors present passed up as *FactorsSoFar*. The Prolog for F112 (for which the acceptance conditions were given earlier, and which will call the procedure to get F111) is:

```
% determine acceptability of
% F112, Information used
getf112(Case,FactorsSoFar):-
member(f18,FactorsSoFar),
write([the,information,was,used]),
nl, getf111(Case,[f112|FactorsSoFar]).

getf112(Case,FactorsSoFar):-
member(f8,FactorsSoFar),
write([the,information,was,used]),
nl, getf111(Case,[f112|FactorsSoFar]).
```

```
getf112(Case,FactorsSoFar):-
member(f7,FactorsSoFar),
write([the,information,was,used]),
nl, getf111(Case,[f112|FactorsSoFar]).

getf112(Case,FactorsSoFar):-
member(f17,FactorsSoFar),
write([the,information,was,not,used]),
nl, getf111(Case,FactorsSoFar).

getf112(Case,FactorsSoFar):-
write([accepted,that,
the,information,was,used]),
nl, getf111(Case,[f112|FactorsSoFar]).
```

Each of the four tests in the acceptance condition is applied in a separate clause, using the set of factors currently identified as present in the case, before proceeding to the next factor (F111), with F112 added to the applicable factors if it is accepted. To allow completion of the database [38], a final clause is added to catch any case not covered by any of the preceding clauses. Although the default is normally *reject*, as discussed above, these defaults may favour either side. In some cases, such as F112 and F104 we decided that the default should be *accept* because in the great majority of the precedents there were no factors in the case descriptions relating to these abstract factors, and yet they are a *sine qua non* for any claim. Our belief is that these aspects were uncontested and so the factors were not explicitly discussed in the trial, and so do not appear in the CATO analysis. Where we felt it was clear that the factor needed to be explicitly established (e.g. F106 (InformationKnown) and F111(QuestionableMeans)) the default was *reject*. The code for F111 is:

```
getf111(C,Factors):-
member(f25,Factors),
write([questionable,means,were,not,used]),
nl, getf123(C,Factors).

getf111(C,Factors):-
member(f17,Factors),
write([questionable,means,were,not,used]),
nl, getf123(C,Factors).

getf111(C,Factors):-
member(f22,Factors),
write([questionable,means,were,used]),
nl, getf123(C,[f111|Factors]).

getf111(C,Factors):-
member(f26,Factors),
write([questionable,means,were,used]),
nl, getf123(C,[f111|Factors]).
```

```
getf111(C,Factors):-
member(f14,Factors),
write([questionable,means,were,used]),
nl, getf123(C,[f111|Factors]).

getf111(C,Factors):-
member(f2,Factors),
  write([questionable,means,were,used]),
nl, getf123(C,[f111|Factors]).

getf111(C,Factors):-
member(f1,Factors),
write([questionable,means,were,not,used]),
nl,getf123(C,Factors).

getf111(C,Factors):-
write([accepted,that,
questionable,means,were,not,used]),
nl, getf123(C,Factors).
```

Note that the use of a default is indicated in the report by the qualification "accepted that".

The above demonstrates that it is a straightforward and reasonably objective process to transform a factor based analysis such as is found in [5] to an executable program via an ADF. Although judgement was sometimes required to form the acceptance conditions, we would suggest that such judgements were not difficult to make. Moreover if there are difficult choices, the effect of the alternatives can be compared on a set of test cases. Producing such alternative versions is a task greatly simplified by the modularisation achieved by use of the ADF. Moreover, the relatively small number of factors relevant to particular nodes further simplifies the task.

#### *4.1.3 Results*

We can now run the program on the cases. We represent the cases as a list of base-level factors. For example, the Boeing case[4] is represented as

```
case(boeing,[f4,f6,f12,f14,f21,f1,f10]).
```

giving output:

```
1 ?- go(boeing).
accepted that defendant is not owner of secret
efforts made vis a vis outsiders
efforts made vis a vis defendant
there was a confidentiality agreement
defendant was on notice of confidentiality
there was a confidential relationship
accepted that the information was used
questionable means were used
```

[4] The Boeing Company v. Sierracin Corporation, 108 Wash.2d 38, 738 P.2d 665 (1987).

```
accepted that the information
    was not available elsewhere
accepted that information is not known
accepted that the information
    was neither known nor available
accepted that the information was valuable
not accepted that the information
    was legitimately obtained
improper means were used
efforts were taken to maintain secrecy
information was a trade secret
a trade secret was misappropriated
find for plaintiff
boeing[f200, f201, f203, f102,f110,f104,f111,
f112,f114,f115,f121,f122,f123,
f4,f6,f12,f14,f21,f1,f10]
decision is correct in accordance with the actual decision
```

The initial program correctly classified 25 out of the 32 cases (78.1%) of the cases. While all ten of the cases won by the defendant were correctly classified, seven of the 22 cases won by the plaintiff were not. The figure for correct answers is remarkably close to the 77.8% reported for the version of CATO used in [34], which, of course, uses exactly the same analysis of the domain and cases that we have adopted here. Thus as a first conclusion we can tentatively suggest that executing the analysis in [5] as an ADF produces very similar results to those obtainable using the original CATO program (albeit we are testing on a smaller set of cases). As Table 2 indicates, the performance of HYPO and CATO is lessened because both make a large number of abstentions. These programs abstain because they lack the precedents which would enable them to decide some novel combinations of factors. IBP (and our program) make far fewer abstentions because the use of the logical model at the top level means that these programs will find for the defendant if they do not find for the plaintiff, forcing a decision. In HYPO and CATO, in contrast, the factors may remain in balance, with arguments for both sides that cannot be conclusively compared.

We would, however, hope to be able to do better than 77.8%, and approach the performance level of IBP. We now, therefore, investigate how the initial program might be improved. The wrongly predicted cases were:

```
case(spaceAero,[f8,f15,f18,f1,f19]).
case(televation, [f6,f12,f15,f18,f21,f10,f16]).
case(goldberg,[f1,f10,f21, f27]).
case(kg,[f6,f14,f15,f18,f21,f16,f25]).
case(mason,[f6,f15,f21,f1,f16]).
case(mineralDeposits,[f18,f1,f16,f25]).
case(technicon,[f6,f12,f14,f21,f10,f16,f25]).
```

Examination of the cases showed that five of the seven had F16 (ReverseEngineerable) present and that these cases were the only cases found for the plaintiff with F16 present. The problem in these five cases is that the program finds for the defendant because the information is available elsewhere (F105). This is established by the presence of ReverseEngineerable and is unchallengeable. Examination of the ADF shows that F16 is immediately decisive: if that factor is present, there is no way the plaintiff can demonstrate that the in-

formation is a trade secret. *Goldberg*[5] also fails through F105 (InformationKnownOrAvailable), since disclosure in a public forum (F27) is also sufficient to deny the information trade secret status. It would appear that we could significantly improve performance by refining this branch to allow the plaintiff some way to defend against, in particular, F16.

#### *4.1.4 Refinement*

At this point we should observe that CATO is likely to be more robust in the face of imperfect analysis than an approach based on a logical model. Because CATO generates arguments based on considering all the available factors taken together, it is less likely to have the outcome determined by a single factor than a logical model, for which, for example, the presence of F16 or F27 can be seen to immediately determine a decision for the defendant, whereas in CATO some combinations of other factors might outweigh them. Moreover, CATO was designed to assist law students, not to predict outcomes. We should expect similar problems to arise in IBP, which also uses a logical model, albeit one that is applied at a later stage of the process. In [34] it is stated

> We found that some Factors, called KO-Factors (or Knockout Factors), almost always dominate the outcome of a case. For instance, as an empirical matter, the plaintiff will not win a case with Factor F20, Info-Known.

Such factors are given special treatment in IBP, and so it does not seem unreasonable to use the initial results to suggest possible refinements to our original analysis. First we consider *Goldberg v Medtronic*. In that decision it is explicitly stated that

> The district court found that Medtronic could not avoid its obligation of confidence due to the availability of lawful means of obtaining the concept when those means were not employed. We affirm.

Thus the factor which was decisive[6] for our progam, F27, was in fact explicitly held to be insufficient in the actual decision. Whether this decision was correct or not is not for us to say, but it does explain why our program misclassifies the case. Assuming the decision to be correct, we should either redefine F27 to include the defendant's actual *use* of this public domain knowledge, so that it is not present in *Goldberg*, or allow F21 (KnewInformationConfidential) as an exception to F27 in determining the acceptance of F106, on the grounds that if the defendant believed the information to be confidential, he could not have been aware that the information was publicly available. Since we have no other case with F21 and F27 both present, we cannot choose between these two solutions on the precedents available to us. We now turn to the problem created by F16, ReverseEngineerable. In [5] the note on the applicability of this factor states:

> The factor applies if: Plaintiff's information could be ascertained by reverse engineering, that is, by inspecting or analyzing plaintiff's product (regardless of whether defendant actually obtained the information in this way).

Thus it is clear that we cannot use a defence analogous to that used in *Goldberg*, that the defendant did not in fact reverse engineer the information. None the less, the ease with which the product could be reverse engineered does (amongst other things) need to be considered. In *Mason* we read:

[5] Goldberg v. Medtronic, 686 F.2d 1219 (7th Cir. 1982).

[6] Running a version of *Goldberg* without F27 finds for the plaintiff.

> In this regard, we note that courts have protected information as a trade secret despite evidence that such information could be easily duplicated by others competent in the given field.

citing *KFC v Marion Kay* and *Sperry Rand v Rothlein*. The KFC decision cited in Mason states

> Marion-Kay maintains that the recipes and formulas for the making of KFC seasoning are not unique and that Marion-Kay is capable, both financially and technically, of producing KFC seasoning.

which suggests that the uniqueness of the product (F15) might be a factor capable of attacking the acceptability of F108 as well as F106 (as identified in CATO). Adding F15 as an exception to F16 would give the correct decision in *Televation*, *KG* and *Mason*. In *Technicon* we find that the phrase “readily ascertainable” is used:

> Curtis claimed that Technicon’s trade secrets were “readily ascertainable” and that the company had not made reasonable attempts to ensure its trade secrets. The Court reasoned that Bridgmon’s "wiretap" process had required over two-thousand hours, and still had not yielded a fully functional product. The Court held that this amount of time indicated that a trade secret was not readily ascertainable.

In fact in two of the cases (*KG* and *Technicon*) restricted material was used by the defendants, strongly implying that the information was not, in fact, readily ascertainable. In *Mineral Deposits*, we find that

> After Zigan received the spiral, he removed the label which indicated that patent applications were pending and gave the spiral to defendant Zbikowski. Zbikowski then cut the spiral into pieces, made molds of the components, and proceeded to manufacture copies of the spiral. If a trade secret is divulged under an express or implied restriction of nondisclosure or nonuse, a party who engages in unauthorized use of the information will be liable in damages to the owner of the trade secret.

which strongly suggests to us that F14 was also in fact present in this case. Moreover in *Televation*, whether the secret counted as reverse engineerable was contested:

> The mere fact, however, that a competitor could, through reverse engineering, duplicate plaintiff’s product does not preclude a finding that plaintiff’s techniques or schematics were trade secrets, particularly where, as here, the evidence demonstrated that the reverse engineering process would be time-consuming.

and there is a strong suggestion that the court in fact believed that copies of the plaintiff’s drawings had, in fact been used by the defendant, which would mean that F14 would apply. Finally *Sperry Rand*, another decision cited in *Mason*, states

> The defendants claim that there is no trade secret if it is disclosed by prior art or if it is readily discernible by others skilled in the field. It is no defense in an action of this kind that the process in question could have been developed independently, without resort to information gleaned from the confidential relationship. As stated in the landmark case of Tabor v. Hoffman, 118 N.Y. 30, 35, 23 N.E. 12, 13 (1889): "Even if resort to the patterns of the plaintiff was more of a convenience than a necessity, still if there was a secret, it belonged to him, and the defendant had no right to obtain it by unfair means, or to use it after it was thus obtained."

which suggests that the use of any kind of questionable means (rather than just F14) could be used to block a defence relying on reverse engineerability. The decisions thus give a number of suggestions for exceptions to F16 as a support for F108: especially uniqueness of the product and use of restricted materials. Incorporating those exceptions would raise success of our program to 29 out of 32 (90.6%), and removing F27 from *Goldberg* (or allowing F21 as an exception) and adding F14 to *Mineral Deposits*, both of which seem eminently justifiable on the facts of the cases concerned as stated in the decision texts, would give correct decisions in these cases also (96.8%). This leaves only *Space Aero* as an unexplained failure. The output for this case is:

```
1 ?- go(spaceAero).
accepted that defendant is not owner of secret
accepted that efforts made vis a vis outsiders
no efforts made vis a vis defendant
accepted that there
     was no confidentiality agreement
accepted that defendant
     was not on notice of confidentiality
accepted that there
      was no confidential relationship
the information was used
questionable means were not used
accepted that the information
       was not available elsewhere
accepted that information is not known
accepted that the information was
    neither known nor available
the information was valuable
not accepted that the information
     was legitimately obtained
accepted that improper means were not used
no efforts were taken to maintain secrecy
information was not accepted as a trade secret
no trade secret was misappropriated
find for defendant
spaceAero[f104,f112,f123,
f8,f15,f18,f1,f19]
decision is wrong
```

This case fails on two branches: the information is not a trade secret because no security measures were taken, and because it appears that no confidential relationship existed. A key feature of this case is that the defendants were former employees of the plaintiff, and had been provided with the disputed information when employed by the plaintiff because they needed it to carry out their duties. The decision itself states

> The testimony, taken as a whole, convinces us that Darling took precautions to guard the secrecy of its process which, under the circumstances, were reasonably sufficient.

This suggests to us that F19 (NoSecurityMeasures) was not, in fact, accepted as present, and removing this factor from the case is enough to establish that there was a trade secret. Turning to the issue of confidentiality we read

> While none of the former employees had signed a contract with Darling in which they formally agreed not to use the information acquired by them, and while they were free to leave their employment at will, Judge Pugh found that they owed the duty of fidelity to their employer while they were employed. We agree. ... The court below found as a fact that some of the former employees had in their possession, after leaving Darling's employment, certain sketches of oxygen breathing hoses which they had taken while they were employed by Darling, without Darling's knowledge. ... the former employees knew that they were acting wrongfully in violation of their confidential relationship and their duty of loyalty. We agree with the court below that the former employees violated the duty of fidelity and trust which they owed to Darling in respect of the trade secret and that their conduct was such as to entitle Darling to the protection of a court of equity.

Again we cannot comment on whether this decision was correctly made or not. However, it does seem that at least F21 (KnewInfoConfidential) should be included. If this is added, we can establish a confidential relationship and find for the plaintiff.

### 4.2 Wild Animals Domain

In the above we have considered the approach with respect to a single domain. If the approach is to be of general significance, however, it needs to be applicable to other domains. In this subsection we will describe a second exercise designed to show that the approach is more generally applicable. We will apply the method to a second domain, one which has often been used as an illustration of factor based reasoning so that analyses are available: the wild animals cases and *Popov v Hayashi*. The wild animals cases were introduced into AI and Law in [23] and extended to the baseball case of *Popov* in [69]. We will use the factor based analysis of [18] as our starting point. Briefly the wild animals cases concern plaintiffs chasing wild animals when their pursuit was interrupted by the defendant. Post was chasing a fox for sport. Keeble was hunting ducks, Young fish and Ghen a whale, all in pursuit of their livelihoods. *Popov v Hayashi* concerned disputed ownership of a baseball (valuable because it had been hit by Barry Bonds to break a home run record). Popov had almost completed his catch when he was assaulted by a mob of fellow spectators and Hayashi (who had not taken part in the assault) ended up with the baseball when it came free. The wild animals cases were cited when considering whether Popov's efforts had given him possession of the ball. Thirteen, base-level, factors are identified in [18]. The first task is to form them (together with appropriate abstract factors) into a factor hierarchy, and to use as the node and link structure to form an ADF. This factor hierarchy is shown in Figure 4; some adaptations to [18] have been made: for example we include a factor *Res* (Residence Status) to indicate the attachment of the animals to the land, since it appears to make a difference (important in some other related cases described in [14]) whether they are on that land permanently, seasonally, habitually, occasionally, or whatever. The nodes and links are given in Table 5. We now supply acceptance conditions for the nine non-leaf nodes.

```
Decide for Plaintiff if NOT (NoBlame)
        AND ((Ownership
            OR (RightToPursue AND IllegalAct))
Ownership if (OwnsLand AND Resident)
        OR Convention OR Capture
Capture if NOT (NotCaught) OR (Vermin and HotPursuit)
```

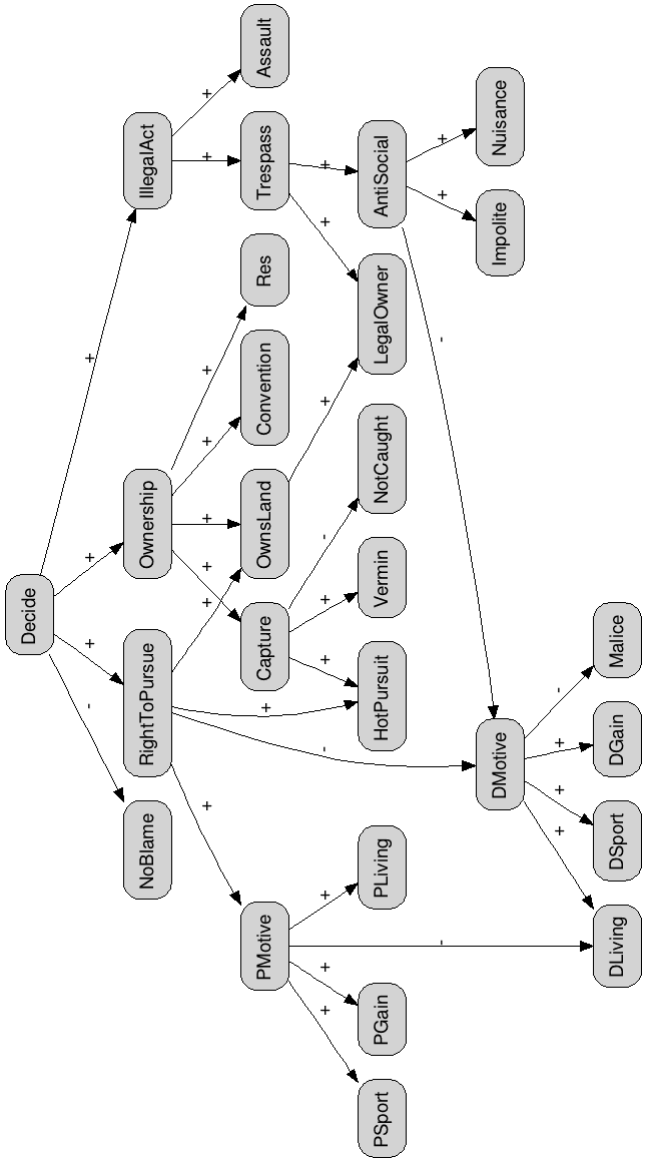


**Fig. 4** Factor Hierarchy/ADF for Wild Animals

```
RightToPursue if OwnsLand OR
           ((HotPursuit AND PMotive
           AND (NOT (better) DMotive))
PMotive if PLiving OR
           ((PSport OR PGain)
           AND (NOT DLiving))
DMotive if NOT Malice AND
           (DLiving OR DSport OR DGain)
```

| S | L+ | L- |
|---|---|---|
| Decide for Plaintiff | Ownership,RightToPursue,IllegalAct | NoBlame |
| Capture | HotPursuit, Vermin | NotCaught |
| Ownership | Convention, Capture, OwnsLand, Res | |
| PMotive | Pliving, PSport, PGain | DLiving |
| DMotive | DLiving, DSport, DGain | Malice |
| OwnsLand | LegalOwner | |
| RightToPursue | OwnsLand, Pmotive, HotPursuit | DMotive |
| AntiSocial | Nuisance, Impolite | DMotive |
| Trespass | LegalOwner, AntiSocial | |
| IllegalAct | Assault, Trespass | |

**Table 5 Wild Animals as ADF**

```
IllegalAct if Trespass OR Assault
Trespass if LegalOwner AND AntiSocial
AntiSocial if (Nuisance OR Impolite)
           AND (NOT DLiving)
```

The only real controversy here is with the determination of RightToPursue when both the plaintiff and the defendant have good motives. Essentially we want to say that if the land is not owned by one of them, the right to pursue is given to the party with the better motive, which will require the some additional code in the implementation to compare these two factors. The remainder seem fairly uncontroversial.

### *4.2.1 Program*

The acceptance conditions can easily be expressed as Prolog procedures and then embedded in code as was done for the CATO program in 4.1.2. We can now execute the program. For example, running the program for *Young v Hitchens* produces the following output (note that the program abbreviates factor names):

```
1 ?- go(young).
the plaintiff had not captured the quarry
the plaintiff did not own the quarry
plaitiff has good motive
defendant has good motive
plainiff did not own the land
plainiff had a right to pursue the quarry
defendant committed no antisocial acts
defendant committed no trespass
no illegal act was committed
do not find for the plaintiff
find for the defendant
young[rtToPursue,dMotive,pMotive,
nc,hp,imp,pliv,dliv]
```

### *4.2.2 Results*

We produce correct results from all five cases discussed in [18], and on this basis we believe that the ADF representation can be used to encapsulate the knowledge of the domain as

represented in [18], suggesting that the method can be applied straightforwardly to a second domain. In general we believe that the method can be applied to any domain for which factor based reasoning in the CATO (or HYPO or IBP) style is appropriate. This has encouraged us sufficiently to apply the method to a larger scale problem in the domain of the US automobile exception to the Fourth Amendment rule for which there is no accepted analysis into factors available, so that we need to start from the case decision texts, as shown in the next section. By starting from the primary texts rather than an analysis, we can also explore the role of the ADF in driving the analysis.

### 4.3 US Automobile Exception To The Fourth Amendment Domain

In this section we apply the approach to ten cases in the domain of the Fourth Amendment, specifically concerning the automobile exception. Although these cases have been discussed in the AI and Law literature (e.g. [62], [43], [21] and [13]), representation of the domain has not been the main focus of these papers and so the analysis performed there has been illustrative rather than complete. In this paper we provide a very detailed analysis, intended to support a complete implementation, and so this represents a substantial test of the use of our methodology on an effectively fresh domain. This example therefore represents an attempt to apply our method starting from scratch, rather than re-using previous analyses.

The Fourth Amendment protects the "right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures." A search is considered reasonable if a warrant has been obtained. However, when there is a high probability of losing the evidence so that there is an urgent reason to search, obtaining a warrant may become impossible. One such situation is a moving automobile. This domain thus considers the interaction of two competing considerations: the enforceability of the law, which makes the exigency issue important, and citizens' rights, which include the right to privacy [13].

This exception was first established by the United States Supreme Court in 1925, in the Carroll v. US. [7] decision which states:

> Various acts of Congress are cited to show that, practically since the beginning of the Government, the Fourth Amendment has been construed as recognizing a necessary difference between a search for contraband in a store, dwelling-house, or other structure for the search of which a warrant may readily be obtained, and a search of a ship, wagon, automobile, or other vehicle which may be quickly moved out of the locality or jurisdiction in which the warrant must be sought.

The automobile exception is developed further as more cases are decided (Table 8 shows the representation of ten landmark cases in the domain) with further conditions needing to be taken into consideration. For example, the type of the vehicle or movable container, the status of the vehicle which influences whether there was an urgent need to search it (e.g. was the vehicle traveling on the highway (as in *Carroll*) or was it parked in a parking lot but capable of moving (as in *California v. Carney* [8])? Was it parked in a private place that is used for accommodation (*Coolidge v. New Hampshire* [9]) and so not subject to inspection without warrant, or was it in a public location? Whatever the situation, there must be a probable cause to search, but is it legal to search the whole vehicle if the probable cause applies only

[7] Carroll v United States, 267 U.S. 132 (1925)
[8] California v. Carney, 471 US 386 (1985).
[9] Coolidge v. New Hampshire, 403 U.S. 443 (1971).

to a container inside the vehicle? What if an authorized warrant was easy to obtain? Such considerations and more are stated in the ADF table shown below. The role of the justices in this domain is to determine whether there is enough exigency with respect to a possibly lowered expectation of privacy given the particular case facts. This is illustrated by the ADF factor hierarchy depicted in Figures 5 and 6 based on the base-level factors given in Table 6.

| ID | Base Factor |
|---|---|
| bf011 | Automobile |
| bf012 | Vessel |
| bf013 | Towable |
| bf014 | LargeContainer |
| bf015 | MovableContainer |
| bf021 | AuthorityOfAvailableMagistrate |
| bf022 | RiskofLosingEvidence |
| bf023 | AvailabilityofMagistrate |
| bf031 | Licence |
| bf032 | RestrictedArea |
| bf041 | OnPublicView |
| bf042 | CanBeSeen |
| bf043 | CanotBeSeen |
| bf051 | UrgentStatus |
| bf052 | CapableToMove |
| bf053 | Public Parking |
| bf054 | PublicLocation |
| bf055 | PermittedDuration |
| bf211 | Information |
| bf212 | Observation |
| bf213 | Procedure |
| bf221 | PublicSafety |
| bf222 | Crime |
| bf231 | WholeVehcile |
| bf232 | OnlyVehicleContainer |
| bf233 | SearchPlace |
| bf311 | GoodsCarried |
| bf312 | ProtectionType |
| bf321 | ConnectedServices |
| bf331 | AccomodationSpaces |
| bf332 | RoomsFunction |

**Table 6 Base-Level Factors in The Automobile Exception as an ADF**

From the case decisions and representations shown in Table 8, we can now generate acceptance conditions for the sixteen non-leaf nodes, using the base-level factors in Table 6. The ADF is given in table form in Table 7.

```
AutomobileException IF Exigency
             AND EnoughExpectationOfPrivacyInUse.

Privacy IF [ EnoughExpectationOfPrivacyInUse
    AND [NOT InspectionRegulation)
         OR (NOT VisibilityofItem)] ]
    OR [PrivateLocation AND (NOT AuthorizedWarrant)].

Exigency IF [(Mobile AND ExigencyWhenApproached
```

| ID | S | L+ | L- |
|---|---|---|---|
| I1 | Exigency | AF202,AF101, AF103,AF104, AF105 | AF102 |
| I2 | Privacy | AF203 | |
| AF202 | ProbableCauseToSearchVehicle | AF121 AF122 AF123 | |
| AF203 | ExpectationOfPrivacyInUse | AF131 AF132, AF133 | |
| AF101 | Mobile | bf011,bf012,bf013,bf014, bf015 | |
| AF102 | EaseWarrant | bf022 | bf021, bf023 |
| AF103 | SubjectToInspectionRegulation | bf031 | bf032 |
| AF104 | VisibilityOfItem | bf041,bf042 | bf043 |
| AF105 | ExigencyWhenApproached | bf051,bf052, bf053,bf054,bf055 | |
| AF121 | AuthorizedOrigionOfProbableCause | bf211,bf212,bf213 | |
| AF122 | UrgentReasonToSearch | bf221,bf222 | |
| AF123 | LegalSearchScope | bf231 | bf232.bf233 |
| AF131 | PrivateContentsCarriage | | bf311,bf312 |
| AF132 | Residence | bf321 | |
| AF133 | Accomodation | bf331,bf332 | |

**Table 7 Automobile Exception as an ADF**

| Case | Base-Level Factors | Favour |
|---|---|---|
| Carroll v. United States | bf011,bf051,bf211,bf231,bf233 | P |
| Chambers v. Maroneys | bf011,bf015,bf051,bf054,bf213,bf222,bf231 | P |
| Coolidge v. New Hampshire | bf011,bf021,bf024,bf051 ,bf213,bf222,bf231 | D |
| Cady v. Dombrowski | bf011,bf042,bf043,bf051,bf053,bf053,bf213,bf222,bf231,bf233 | P |
| South Dakota v. Opperman | bf011,bf015,bf043,bf051,bf053,bf213,bf221,bf231,bf233 | P |
| United States v. Chadwick | bf011,bf014 ,bf043,bf051,bf053,bf211,bf221,bf232,bf233,bf312 | D |
| Arkansas v. Sanders | bf011,bf014,bf015,bf032,bf043,bf051,bf211,bf221,bf232 | D |
| United States v. Ross | bf011,bf015,bf032,bf051,bf053,bf211,bf221,bf231,bf232,bf233, bf233,bf311 | P |
| California v. Carney | bf011,bf015,bf022,bf031,bf032,bf051,bf052,bf053,bf054,bf211, bf212,bf221,bf231,bf233,bf233 ,bf331,bf332 | P |
| California v. Acevedo | bf011,bf015,bf032,bf043,bf051,bf054,bf211,bf221, bf232,bf233 | P |

**Table 8 Automobile Exception Cases**

```
        AND ProbableCauseToSearch)
        OR (NOT EaseToGetWarrant)].

EnoughExpectationOfPrivacyInUse IF
                    (Residence AND Accommodation)
                     OR PrivateContentsCarriage.

Residence IF ConnectedMainLivingServices.

PrivateContentsCarriage IF GoodsCarried
                  AND ProtectionLevel AND ContainerType.
```

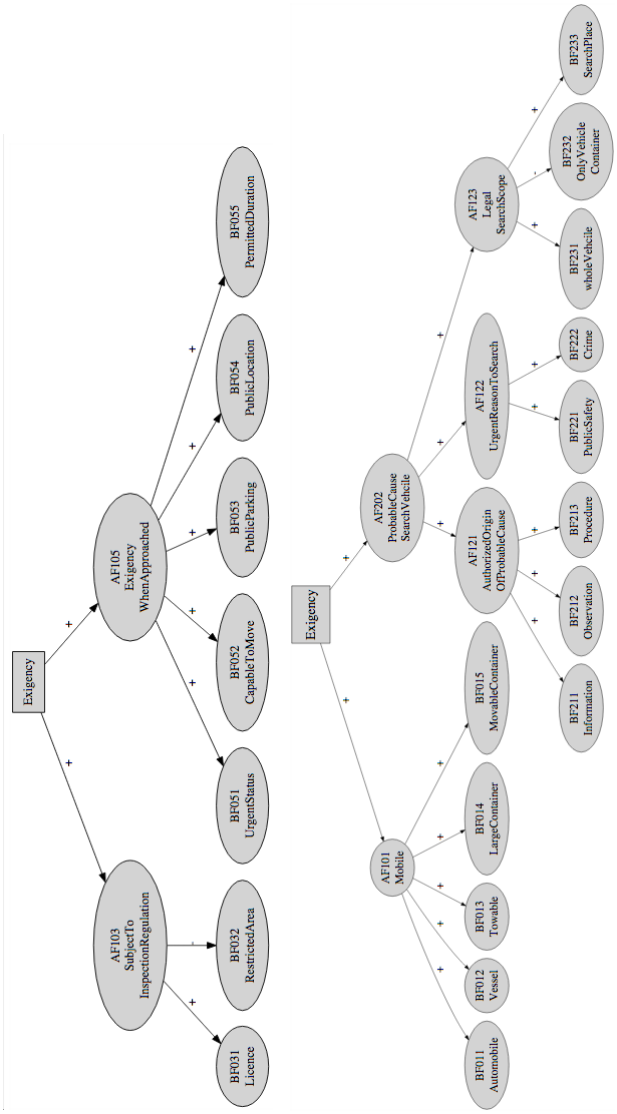


**Fig. 5** Factor Hierarchy/ADF for Automobile Exception, part 1

```
Accommodation IF AccommodationSpaces OR RoomsFunctions.

SubjectToInspectionRegulation IF License
                    AND (NOT RestrictedArea).

VisibilityofItem IF OnPublicView OR CanBeSeen
                   OR (NOT CannotBeSeen).
```

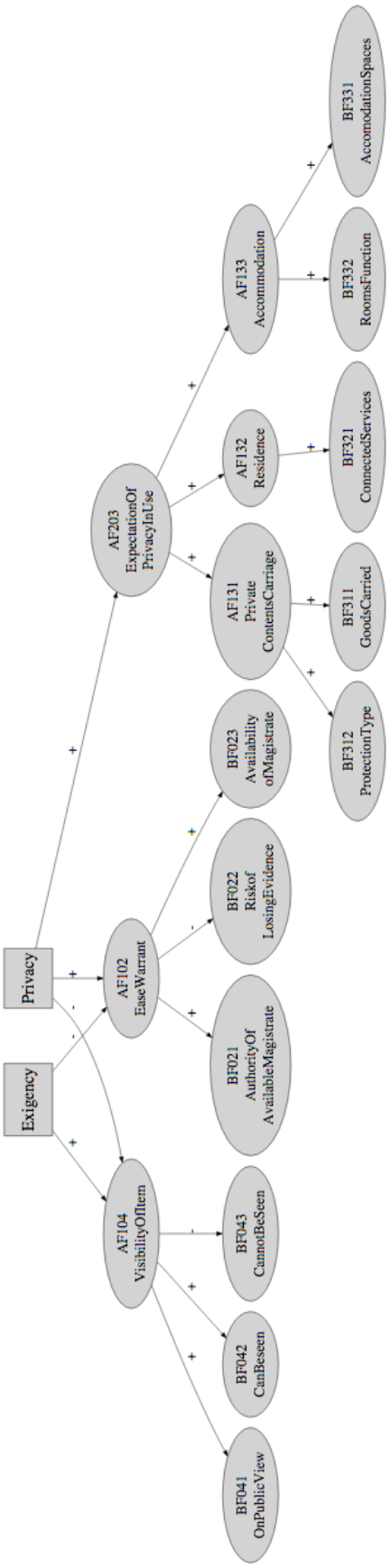


**Fig. 6** Factor Hierarchy/ADF for Automobile Exception, part 2

```
ExigencyWhenApproached IF (UrgentStatus AND PublicLocation)
      OR (CapableToMove AND
          (PublicLocation OR PublicParking
           OR PermittedParkingTime)).

Mobile IF Automobile OR Vessel
       OR TowableVehicle OR LargeContainer
       OR MovableContainer.

EaseOfObtainingWarrant IF (NOT RiskofLosingEvidence)
         OR (Magistrate availability
             AND AuthorityOfMagistrate).

ProbableCausetoSearchVehicle IF OriginPurpose
              AND LegalUrgentReasonToSearch
              AND LegalSearchScope.

OriginPurpose:= Information OR Observation
                            OR Procedure.

UrgentReasonToSearch:= PublicSafety OR Crime.

LegalSearchScope:=(WholeVehicle OR VehicleContainer)
            AND LegalSearchPlace.
```

For each of the acceptance conditions described above, we define a *default* value when none of the tests apply. The default value is determined with respect to the nature of the factor: for example, the content is not considered private if none of the acceptance conditions for (AF131-PrivateContentsCarriage) are satisfied, and there are only two reasons which can make a search urgent.

### *4.3.1 Program*

The acceptance conditions are implemented using Prolog procedures as in our previous domains, and ordered to ensure that we provide a report giving the status of every non-leaf factor in the domain. The following procedure shows the code for the two acceptance conditions and the default clause for AF202-ProbableCauseSearchVehicle.

```
getProbableCauseSearchVehicle(C,Factors):-
member(af123,Factors),
member(af122,Factors),
member(af121,Factors), !,
write([there, is, a ,probable, cause, to , search,
vehicle ]),nl,getSubjectToInspection(C,[af202|Factors]).

getProbableCauseSearchVehicle(C,Factors):-
member(af122,Factors), member(af121,Factors), !,
```

```
write([there, is, a ,probable, cause, to , search,
vehicle , but, the , search, scope, was , illegal ]),
nl,getSubjectToInspection(C,Factors).

getProbableCauseSearchVehicle(C,Factors):-
!,write([default,there, is, no, probable, cause, to ,
search, vehicle ]),nl,getSubjectToInspection(C,Factors).
```

The output below is for *California v Carney* (abbreviated in the code to"cvc"). This case has frequently been used in AI and Law to explore Supreme Court oral argument (e.g. [62], [3]). Carney is concerned with whether the exception for automobiles to the protection against unreasonable search provided by the Fourth Amendment applies to mobile homes, in particular motor homes in which the living area is an integral part of the vehicle (i.e what is often called a "camper van" as a opposed to a "caravan", in which the living accommodation is separate from the towing vehicle). The decision held that the exception does apply:

> When a vehicle is being used on the highways or is capable of such use and is found stationary in a place not regularly used for residential purposes, the two justifications for the vehicle exception come into play. First, the vehicle is readily mobile, and, second, there is a reduced expectation of privacy stemming from the pervasive regulation of vehicles capable of traveling on highways. Here, while respondent's vehicle possessed some attributes of a home it clearly falls within the vehicle exception.

Thus the case decision indicates that although the mobile home is an automobile with an accommodation space, it was not parked in a residential parking and not connected to any services, and so was currently being used as a vehicle not a home. There was a probable cause to search the mobile home to protect the public after police agents observed suspicious activity, and it was not possible to obtain a warrant since the vehicle was able to move, and the highway was readily accessible from the parking lot.

```
case(cvc,[bf011,bf015,bf022,bf031,bf032,bf051,
          bf052,bf053,bf054,bf211,bf212,bf221,
          bf231,bf233,bf233 ,bf331,bf332]).

?- go(cvc).
[it,is,mobile]
[there,was,exigency,when,approached]
[there,was,an,authorized,origin,of,probable,cause]
[the,main,reason,to,search,was,urgent]
[the,search,scope,is,legal]
[there,is,a,probable,cause,to,search,vehicle]
[subject,to,regular,inspection,but,
    the,search,was,directed,at,restricted,area]
[accepted,that,it,is,not,visible,to,public]
[accepted,that,contents,are,not,considered,private]
[accepted,that,it,is,not,connected,to,one,
      or,more,main,living,services]
[the,place,could,be,used,for,accommodation]
[accepted,that,low,expectation,of,privacy,in,use]
```

```
[it,is,not,easy,to,obtain,warrant]
[justified,under,automobile,exception]
[reduced,expectation,of,privacy]
[warantless,search,did,not,violate,the,fourth,amendment]
found for the plaintiff
```

Another earlier case, US. v Chadwick[10], provides further details for the automobile exception in terms of the part of the vehicle that could be searched. *Chadwick* was found for the defendant so that the exception did not apply. Here, the agents searched a double-locked footlocker placed in the car trunk of a parked automobile without obtaining a warrant. The decision states that even if there was an urgent situation since the vehicle could drive away, the probable cause arises only from the footlocker which should have been seized (before being placed in the vehicle, which was perfectly possible) but not searched until a warrant had been obtained. The case output indicates that the warrantless search here violates the Fourth Amendment rule.

```
case(usvc,[bf011,bf014 ,bf043,bf051,bf053,bf211,bf221,
           bf232,bf233,bf312]).
|    go(usvc).
[it,is,mobile]
[there,was,exigency,when,approached]
[there,was,an,authorized,origin,of,probable,cause]
[the,main,reason,to,search,was,urgent]
[the,search,scope,is,illegal]
[there,is,a,probable,cause,to,search,vehicle,
            but,the,search,scope,was,illegal]
[subject,to,regular,inspection]
[accepted,that,it,is,not,visible,to,public]
[accepted,that,contents,are,not,considered,private]
[accepted,that,it,is,not,connected,to,
           one,or,more,main,living,services]
[accepted,that,the,place,is,not,used,for,accommodation]
[accepted,that,low,expectation,of,privacy,in,use]
[it,is,not,easy,to,obtain,warrant]
[reduced,expectation,of,exigency]
[intrusion, on,privacy,is,not,justified,
            under,automobile,exception]
[warantless,search,violates,the,fourth,amendment]
found for the defendant
```

### *4.3.2 Results and Refinements*

The program output shows that 9 of the 10 cases in this domain are decided correctly by our program. Only the most recent case in the set (*California v Acevedo*[11]) was decided wrongly. The Justices saw *Acevedo* as clarifying some doubtful findings in earlier decisions

[10] United States v Chadwick, 433 U. S. 1 (1977)
[11] California v. Acevedo, 500 U.S. 565 (1991)

(in particular *Chadwick* and *Sanders*). In *Acevedo*, searching the vehicle at the police station, when the probable cause arose only from a container in the trunk, without obtaining a warrant was held to be legal under the automobile exception to the Fourth Amendment rule. This is despite the apparent precedents of *Chadwick* and *Sanders*.

> Separate doctrines have permitted the warrantless search of an automobile to include a search of closed containers found inside the car when there is probable cause to search the vehicle, United States v. Ross,456 U. S. 798, but prohibited the warrantless search of a closed container located in a moving vehicle when there is probable cause to search only the container, Arkansas v. Sanders,442 U. S. 753. Pp. 500 U. S. 569-572.

This illustrates an example where citing over-ruled case decisions produces a wrong result (the domain has entered the third stage of Levi's life cycle). The previously accepted rule is explicitly rejected by the court in the *Acevedo* decision:

> The Chadwick-Sanders rule also is the antithesis of a clear and unequivocal guideline and, thus, has confused courts and police officers and impeded effective law enforcement

The program cannot, of course, detect that an existing understanding of the precedents is about to be changed, and so the output of *Acevedo* follows the *Chadwick* decision and indicates, wrongly, that the search is not justified under the automobile exception.

```
case(cva,[bf011,bf015,bf032,bf043,bf051,bf054,bf211,
          bf221,bf232,bf233]).

 ?- go(cva).
[it,is,mobile]
[there,was,exigency,when,approached]
[there,was,an,authorized,origin,of,probable,cause]
[the,main,reason,to,search,was,urgent]
[the,search,scope,is,illegal]
[there,is,a,probable,cause,to,search,vehicle,but,the,search,
                       scope,was,illegal]
[subject,to,regular,inspection,but,the,search,was,directed,
          at,restricted,area]
[accepted,that,it,is,not,visible,to,public]
[accepted,that,contents,are,not,considered,private]
[accepted,that,it,is,not,connected,to,
              one,or,more,main,living,services]
[accepted,that,the,place,is,not,used,for,accommodation]
[accepted,that,low,expectation,of,privacy,in,use]
[it,is,not,easy,to,obtain,warrant]
[reduced,expectation,of,exigency]
[intrusion, on,privacy,is,not,justified,
                under,automobile,exception]
[warantless,search,violates,the,fourth,amendment]
found for the plaintiff
```

The program here produces a wrong decision giving the priority to the Chadwick-Sanders rule, whereas the actual decision is that the case should fall under the automobile exception

rule if there is a probable cause to search a container inside a vehicle and the circumstances mean that no warrant is required.

> Police, in a search extending only to a container within an automobile, may search the container without a warrant where they have probable cause to believe that it holds contraband or evidence. *Carroll v. United States*.

A number of possible refinements could be applied to resolve such a problem. The program can be adjusted in terms of adding new facts to represent the degree of acceptance of the base-level factor, or by using a portion of precedent cases as discussed in the next section. Essentially this decision is intended to initiate a new period of settled laws, in which *Chadwick* and *Sanders* do not have force, whereas the program represents the pre-*Avecedo* situation. Of course a factor based program such as ours cannot get both *Chadwick* and *Avecedo* correct, since they follow different understandings of the law. To decide both cases correctly would require the inclusion of a temporal context. This has been quite widely discussed with respect to statutes, e.g. [57], but there is very little discussion with respect to cases, although the issue was raised in [24] and the effect of cases appearing in different sequences explored in [46]. In consequence we do not feel that the temporal context is sufficiently well understood to form part of a methodology, and so we leave temporal considerations to one side for the present.

## 5 Discussion

This section evaluates the approach of using ADFs to design and implement a program to decide legal cases based on knowledge derived from a number of precedents in a particular case law domain. As shown by the examples in the previous section, we start by analysing a number of decided cases. After that, we construct the ADF from the factors and issues, showing the support and attack relations between parents and their children and having the base-level factors as the leaves of the ADF. We define the acceptance conditions for each node, translate them to Prolog procedures, run the program against a set of test cases and compare the decisions from the program output to the actual outcomes. From the results obtained above, we can state a number of findings, as given below:

### 5.1 Comparative Evaluation

- Applying the approach on three domains shows the effectiveness of the method for encapsulating the theory developed in the analysis.
- The previous section shows that by using the ADF we can readily explain the points at which the acceptability conditions do not concur with the decisions taken in the actual cases when using a set of factors identified for the case. We can then return to the original decisions and use them to determine possible refinements to the representation of the case, or the domain knowledge. In some cases, such as those we encountered in the US Trade Secrets domain, the problem seems to lie with the attribution of the factors. Should *Goldberg* really have F27? Should *Mineral Deposits* have F14? Should *Space Aero* include F14 or F21 and exclude F19? Such matters were contested in the actual case, and ascribing the presence or absence of particular factors requires interpretation of the case by the analyst.

- As well as disputed factors, a decision like *Goldberg* suggests that we may wish to modify the description of factors intended to guide the analyst. In that decision it was suggested that to count for the defendant, the information not only had to be publicly available but that the public source needed to be known and *used by the defendant*, which would narrow the applicability of F27 as described in [5].
- Adding or removing a factor to or from a particular case provides a local solution which will often solve a problem with a particular case. Our results, however, indicated a general problem which was applicable to several cases: In the Trade Secrets domain F16, reverse engineerable, had a dominant effect, which led to an incorrect decision in several cases. It seemed clear to us that the presence of F16 should not by itself be sufficient for a finding for the defendant. Again, the decisions themselves suggested several possible ways of arguing against F16: in particular the use of restricted materials and the uniqueness of the product. Either or both of these exceptions could be incorporated into the ADF without adversely affecting the other cases.
- Finally it should be conceded that the decisions themselves may be erroneous. Assuming that there are least some poor decisions which we would not wish to serve as precedents, we should be willing to tolerate a certain number of divergences from our results. Moreover, a landmark case like *Avecedo* may significantly change the interpretation of some previous decisions, revising our understanding of the applicable case law and necessitating revision of the program.

To summarise:

- Analysing a legal domain by simply translating the analysis of [5] into an ADF and executing the resulting program gave results almost identical to those found for CATO in the IBP experiments reported in [34]. Note that this is achieved without need for balancing of pro and con factors central to existing case based reasoning systems.
- The reasons for the "incorrect" decisions can be readily identified from the output and the ADF, as we saw from the discussion of the wrongly decided cases in US Trade Secrets and Automobile Exception domains.
- Examination of the texts of the decisions readily explained why the results diverged, and suggested ways in which the analysis could be improved, either at the case level by changing the factors attributed, or at the domain level by including additional supporting or attacking links.

From this we conclude that use of ADFs provides good performance, and has a number of positive features from a software engineering (and domain analysis) standpoint, which enable the ADF to be refined where needed and performance improved. For a practical system there are often several ways to fix a problem, and we would need to have a reasonably large set of test cases in order to choose between the different solutions and to guard against over-fitting.

### 5.2 Values and Teleological Aspects

Note that the ADF structure, like CATO and IBP, does not include any reference to values. Since [23] it has become usual to resolve conflicts not definitively decided in the precedent base by appealing to the purposes of the law, commonly represented as values[12], as in [15]

[12] In the sense of [17].

and other related work. The relation between ADFs and values was discussed in [2], but merits discussion here also.

In [15] values appear at the top of the hierarchy, occupying the place taken up by issues in the representation described above. The idea is that factors are associated with values, and that the precedents, by showing preferences between sets of factors, will reveal preferences over values. Since there are more factors than values, and several factors relate to the same value, these value preferences can be used to determine preferences between sets of factors which do not themselves appear in the precedents, allowing the system to draw conclusions that go beyond *a fortiori* reasoning. The use of values has been criticised and does not appear in any of HYPO, CATO or IBP. In AGATHA [37], which was intended as an empirical evaluation of the approach of [15], the issues of IBP were used in the role of values. Our view here is that values should not form part of the structure: their role is to justify components of the structure and choices made. They thus should form part of the documentation of the design rather than part of the design itself. Value might therefore, for example, appear as comments on the acceptance conditions explaining why the tests are ordered as they are. As observed in [70] values do not play a single role: they can, for example, justify either the inclusion of a rule, or of an antecedent in the rule. In ADF terms, this means that they can justify the inclusion of a test within an acceptance condition, or a node representing a factor to be used in such tests. Moreover, playing the role assigned to them in [23] and [15] they can explain why the various tests in acceptance conditions are ordered in a particular way. Thus the knowledge engineer will need to identify a set of applicable values, and an ordering on them, to guide the design choices, but they will not form an explicit part of the ADF, just as their appearance in actual decision texts is very limited. Note too that since the value preferences are now effectively local to particular nodes, it is possible to express different preferences in different contexts, which may add a desirable flexibility. If it is considered undesirable, the knowledge engineer must ensure, as part of the verification activity, that the preferences are, in fact consistent.

A means of extracting values from an ADF, from associations between values and factors, was given in [2]. In our current methodology, however, values inform the design but do not form part of it. As will be mentioned in our concluding section, it might in future prove useful to provide links to values to improve explanations, but this will affect only presentation, not the decision making, since the effect of values and preferences to them is already cashed out in the representation, in the form of the components included, the tests in the acceptance conditions and the order of these tests.

### 5.3 Quality of Explanations

As the Prolog program proceeds it reports on the acceptability or otherwise of the various abstract factors and the resolution of issues. As shown above, this provides an excellent diagnostic for divergent decisions, but how does it measure up to the actual decisions found in cases? Of course, without facts, we will not be able to follow the decision very closely. But, consider a re-ordering of the elements of our decision for, say *Boeing*. We also omit some elements, and add a little linking text. Recall too that we wrote the program used thus far to "decide" the cases: in a version to supply explanations we would want to customise the text reports to indicate the particular clause being used for a node by giving the base level factors used. Below is what a decision might look like: we show the current program output in boldface, possible clause-specific customisations in italics and linking text in ordinary font.

> We **find for plaintiff**. The **information was a trade secret**: **efforts were taken to maintain secrecy**, *since disclosures to outsiders were restricted and the defendant entered into a non-disclosure agreement and other security measures were applied.* **The information was unique**. It is **accepted that the information was valuable** and it is **accepted that the information was neither known nor available**. **A trade secret was misappropriated**: **there was a confidential relationship** *since the defendant entered into a non-disclosure agreement* and it is **accepted that the information was used**. Moreover **improper means were used** *since the defendant used restricted materials*.

This seems to have the makings of a reasonable summary of the decision. There are two problems: it does not indicate what the defendant contended, since the clauses of the program which were not reached do not feature in the report, and, of course, the facts on which the finding are based are not present. None the less, we find the output a distinct improvement on previous work such as [36]. We believe that the output from the current program could be readily used to drive a program of the sort envisaged by Branting [28], and that this will become even more so if a fact layer to allow the explanation of the attribution of factors is added. Integration with facts is considered in section 5.3. What is also missing, however, is the citation of precedents which are such an important feature of real decisions: we will consider this aspect in section 5.4.

### 5.4 Facts Layer

We believe that we do need to include a fact layer to increase transparency in the ascription of base factors to cases. The interpretation of the cases cannot be disputed without descending to the level of facts as advocated by [11] and [1]. Each legal case should be represented as a set of facts that determine the base factors applicable to the case and thus allow the explanation of the attribution of factors, which in turn provides the basis for deciding the case using a set of abstract factors to resolve the conflict in the issues.

For example, let us again consider here *Carney*, from the domain of Automobile Exception. We will redefine *Carney* using the facts mentioned in the decision of the case rather than the base-level factors and extend the program to provide acceptance conditions for the base factors using a number of possible facts. These facts are determined from the main automobile exception rule, such as the type of vehicle (e.g. car, ship, wagon), personal effects (e.g. paper bag, suitcase), private place (e.g. store, dwelling-home). Additional facts are added to provide similarity and/or differences from precedent cases, or to show the effect of inserting new facts, such as mobile home in *Carney*, on the decision of the case. The output below reports the decision for *Carney* after incorporating the case facts:

```
case(cvc,[ft011mh,ft015pb,ft022nc,ft031mh,ft032ps,
        ft051p,ft052dr,ft053pl,ft054d,ft211pi,
        ft212po,ft221is,ft231all,ft233ps,ft233al,
        ft311is,ft312c,ft331c,ft331as,ft332bd,ft332k]).

1 ?- go(cvc).
[it,is,a,mobileHome,vehicle]
[accepted,that,no,largeContainers]
[paper,bag,is,a,movableContainer]
[it,is,mobile]
```

```
[there,was,no,urgent,status,automobile,was,parked]
[accepted,that,the,vehicle,is,capable,to,move]
[the,vehicle,was,in,public,location]
[the,vehicle,was,parked,in,public,parking]
[accepted, that,the,vehicle,was,parked,for,unknown,period]
[there,was,exigency,when,approached]
[received,information,from,public,informant]
[there,was,an,authorized,origin,of,probable,cause]
[main,reason,to,search,was,to,protect,the,public]
[the,main,reason,to,search,was,urgent]
[all,vehicle,parts,have,been,searched]
[the,vehicle,was,searched,twice,at,the,original,
        automobile,location,
         and,at,police,station]
[the,search,scope,is,legal]
[there,is,a,probable,cause,to,search,vehicle]
[has,a,special,motorhome,licence]
[police,station,is,a,restricted,area]
[subject,to,regular,inspection,but,the,search,was,directed,at,
           restricted,area]
[accepted,that,item,is,not,on,public,view,or,details,are,not,provided]
[accepted,that,item,can,not,be,seen,by,public,
                or,details,are,not,provided]
[accepted,that,it,is,not,clear,that,items,can,not,be,seen]
[accepted,that,it,is,not,visible,to,public]
[illegal,goods]
[just,closed,but,not,protected]
[accepted,that,contents,are,not,considered,private]
[accepted,that,none,of,living,main,services,are,
                       specified,or,connected]
[accepted,that,not,connected,to,
              one,or,more,main,living,services]
[consists,of,a,cab,and,suitable,accommodation,space]
[the,place,could,be,used,for,accommodation]
[accepted,that,low,expectation,of,privacy,in,use]
[accepted,that,there,is,risk,to,lose,evidence]
[accepted,that,magistrate,is,not,available]
[accepted,that,authorized,magistrate,is,not,available]
[it,is,not,easy,to,obtain,warrant]
[justified,under,automobile,exception]
[reduced,expectation,of,privacy]
[warantless,search,did,not,violate,the,fourth,amendment]
found for the plaintiff
```

In comparison to the output of the same case presented in section 4.4.1, we can clearly observe the improvement in the quality of the explanation that provides the justification for the acceptance of the base-level factors. The order of the output states the status of the base-level factors first (children) and then indicates whether the abstract factor (Parent of

base-level factors) is satisfied or not. Now following the approach discussed in the previous section, we can produce *Carney's* decision from the program report:

> The **warantless search did not violate the Fourth Amendment** rule. The respondent's **mobile home** was **capable to move** but found **parked** in a **public location**.The two **justifications for the vehicle exception** come into play. First, the vehicle is **readily mobile** and there is a **reduced expectation of privacy**.
> While the **mobile home** here **consists of cab and accommodation space**, it is **subject to inspection under vehicle regulation**. Also, there was a **probable cause to search the vehicle** based on the fact that the *respondent* was using the **mobile home** to sell **illegal contraband**

This decision now is very close to the actual decision. It clearly justifies what base-level factors have been accepted (or rejected). Moreover, considering the case facts can provide further advantage in specifying a degree to what level the base factor is satisfied.

Further discussion of the inclusion of a facts layer can be found in [4]. There the organisation of facts using HYPO-like dimensions is advocated. Dimensions are also used as a way of moving from facts and factors in [11], [7] and [8]. The use of dimensions does indeed seems a promising way forward, but we feel that the use of dimensions is not yet as well understood as the use of factors, and so more investigation will be needed before it can form part of a generally applicable methodology for the design and development of substantial systems. For this reason we have represented cases as bundles of factors in the manner of [5], [60], [15] and [47], leaving the extension to facts, perhaps using dimensions, for future work.

### 5.5 Portions of Precedents

Using the ADF approach, we do not see confrontations between large sets of pro and con factors covering the whole case. Instead factors are opposed to one another in the context of accepting or rejecting particular nodes, and so will represent a specific point in the argument. Thus two cases may be identical with respect to a subset of factors, which may be used to establish a particular abstract factor, even though the eventual outcome may differ.

Two points are significant here: first that some apparent distinctions are not significant, since they relate to different issues; this was partly what the factor hierarchy was introduced in CATO to address. Here the requirement for significance is even more fine grained, being applicable at the abstract factor level. More importantly, however, a precedent might be citable to establish the existence of a certain plaintiff factor, even though the case as a whole was found for the defendant, because of some other issue. For this reason it is sometimes desirable to be able to reason with portions of precedent, as urged by Branting in [27]:

> This paper argues that the task of matching in case-based reasoning can often be improved by comparing new cases to portions of precedents. An example is presented that illustrates how combining portions of multiple precedents can permit new cases to be resolved that would be indeterminate if new cases could only be compared to entire precedents.

This is borne out by a reading of the decisions in the various cases: rarely do they begin with a precedent and then discuss similarities and differences, but rather they use precedents at particular points of the decisions to identify questions and issues to be addressed, and to justify answers and consequences. This is effectively what is done in the ADF approach:

competing factors are considered in the context of accepting or rejecting a particular node. Part of the output of the program is, for each case, the set of nodes satisfied. This information could be used to find the precedents needed to make particular points.

For example, in US Trade Secrets, we could retrieve all cases where the defendant had agreed not to disclose (F4) and yet efforts to maintain secrecy (F102) were not established. This query would return *CMI* where the information was known to competitor (F20), and so we can cite *CMI* as a precedent when arguing (in the context of a case containing F4 and F20) that the efforts taken to maintain secrecy were insufficient to establish the information as a trade secret. In addition, the decision in *California v. Acevedo*, from the Automobile Exception domain, cites some precedents that support the defendant, but, does not follow these citations due to the perceived lack of clarity of the decision in the precedents. Thus, it is important to clarify these citations in the output of the program in order to provide more transparency and closeness to the actual decision.

Matching at this level of granularity will direct us to the precedents most relevant to the specific point we need to argue. Moreover, such precedents can then be incorporated in our explanations to justify the acceptance or rejection of particular nodes, which is close to the way they are used in practice, and which moreover corresponds to the downplaying and emphasis of distinctions which are such an important feature of CATO. To do this it would be helpful to be able to argue about preferences (perhaps using some form of meta-level argumentation as in [20]).

Another reason to consider portions of precedents is provided by [47]. In that paper *a fortiori* reasoning was explained in terms of identifying a rule using only a subset of factors available for the winning side preferable to the rule using all the factors available to the losing side. That paper gave, however, no indication of how this subset should be chosen. The output from the ADF in contrast, does show which factors were instrumental and active in winning the case.

## 6 Conclusion and Future Work

In this paper we have described a new and promising approach to the design and implementation of systems to suggest precedent based decisions to legal cases using Abstract Dialectical Frameworks, a recent development in computational argumentation. We have shown for several legal domains how ADFs can be used to encapsulate a theory of knowledge of their case law, in order to form the basis of a design of a program to decide cases in those domains. The theories are constructed by the domain analyst through expressing the cases in terms of factors and implemented taking advantage of the closeness of the representation to an executable form by translating the acceptance conditions of the ADF into Prolog procedures. By evaluating the approach on these three domains, we conclude that:

- The success of the approach depends to a large extent on the quality of the analysis. It is important that the domain modelled be in a stable state (the second stage of Levi's life cycle explained in [53]). However, the ADF design facilitates refinement of the program in the light of its performance on test cases, to remedy defects in the initial analysis.
- The method can be applied to different domains where factor based representation is considered an appropriate approach to capture the knowledge of the domain. Normally, the results are affected by the purpose of the analysis and the size of the knowledge. US Trade Secrets required more refinements as that has been translated from the analysis of CATO which was designed to perform a different task, namely to identify arguments rather than to predict outcomes.

- The modularisation achieved by using the ADF approach for design can not only help drive the analysis, but also provides a number of software engineering benefits, such as ease of following the flow of control through the knowledge bases, limitation of the effects of modifications, and assisting the quality assurance, verification and validation of the program built upon the design.
- The ADF provides a way of mediating between frame and rule based representations of the domain knowledge.
- Programs based on the ADF design provide very transparent output that identifies precisely where the outcomes suggested by the implementation diverge from the actual outcomes. After running the program, and noting any divergences from the actual outcomes, reading the original decision texts can suggest one of four solutions to refine the behaviour of the program. These are, in ascending order of divergence from the original analysis:
  - Removing a factor wrongly attributed to the case
  - Adding a factor wrongly omitted from the case
  - Modifying an acceptance condition: e.g. changing the priorities
  - Modifying the ADF: e.g. adding a supporting or attacking node for the problem node.

  Often several of these modifications can potentially solve the problem, and the choice is made according to the context provided by the other divergent cases we are trying to accommodate.

We find all of this encouraging and offering further pointers as to how to address important issues in future work. Thus, the next technical step will be to supply the fact layer in a more complete way than was sketched in section 5.3. This will require further analysis of the original decisions and oral transcripts to extract facts, so as to permit argument about the ascription of factors, and to enable grounding of our explanations in the particular facts of a case. Our current thinking, expressed in [4], is that reviving the idea of dimensions will provide a good way of tackling this issue, as indicated by [11] and [8]. Descent to the fact layer will result in a considerable improvement in the transparency of the output by enabling consideration of important features from the actual case decisions. Once the method has been extended to include the facts of particular cases at the lowest level of the ADF, a program to present the output in a form resembling the texts of actual decisions can be considered. This will need some post-processing such as that found in [28].

Much of the existing work depends critically on the ascription of factors to cases in a Boolean manner, while in practice there are compelling reasons to see the presence of factors as a matter of degree. On the basis of our observations we believe that considering the fact layer in the ADF will help in providing answers to determine the values of these degrees taking advantage of the weighted links supported by ADFs [33]. We will start from the approach suggested in [2]. Assigning degrees to justify the presence of the base factors will affect all the parents in the ADF and thus provide more flexible justification of the resolution of the conflicting issues. Preliminary efforts in this direction, exploiting dimensions, are reported in [4].

In addition to the facts determining degrees of presence for factors, we also need to improve the transparency of the reasoning by justifying the preferences between portions of precedents, which better corresponds to legal practice as manifest in real decisions and may express social values and judicial preferences. Ideally each test in the acceptance condition should be related to a precedent case; this will require annotating the acceptance conditions with citations to precedent cases, which will in turn improve the program output. This could

also provide a potential database to support conceptual retrieval of cases, which has been an important issue in AI and Law since its very beginnings (see, e.g. [45]).

Also worthy of exploration is producing a variant program to generate all the possible arguments for the majority, minority or concurring arguments required to resolve conflicts explicitly, so as to facilitate comparison with ASPIC+. Our Prolog program directly implements the acceptance conditions for each node in the ADF, and, for a given set of facts, the program always produces a single outcome. This is to be expected given the absence of cycles in our ADF, which capture the tree-based structure of CATO's factor hierarchy. In future work, we intend to take advantage of the Diamond system [41] which directly implements ADFs and can produce extensions corresponding to a variety of semantics. For example, preferred semantics may be helpful for capturing variant opinions in a case.

Aside from these technical aspects, we recognise that our examples have all been taken from the AI and Law literature, and even though we have included what is probably the most substantial analysis to date (CATO/IBP), the examples are still relatively small. The true test of a methodology and its supporting tools comes from its use in practice, and so we would hope that our proposal would be adopted by people wishing to build a substantial system to perform factor based reasoning in a legal domain. Such a tool cannot be the subject of a research project carried out by academic computer scientists: it requires the motivation and personnel that only a practical project can provide. As practical applications appear to be moving closer[13], we hope and expect the methodology will be used and proven in practice.

## Acknowledgements

Would like to thank the anonymous reviewers of the first version of this paper, whose comments have helped us to clarify our ideas and significantly improve the paper.

## References

1. L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. Abstract dialectical frameworks for legal reasoning. In *Proceedings of Jurix 2014*, pages 61–70, 2014.
2. L. Al-Abdulkarim, K. Atkinson, and T. Bench-Capon. Factors, issues and values: Revisiting reasoning with cases. In *Proceedings of the 15th ICAIL*. ACM, 2015.
3. L. Al-Abdulkarim, K. Atkinson, and T. J. M. Bench-Capon. From oral hearing to opinion in the us supreme court. In *Proceedings of JURIX 2013*, pages 1–10, 2013.
4. L. Al-Abdulkarim, K. Atkinson, and T. J. M. Bench-Capon. *Adding Dimensions and Facts to ADF Representation of Legal Cases*. University of Liverpool Department of Computer Science Technical Report ULCS-15-004, 2015.
5. V. Aleven. *Teaching case-based argumentation through a model and examples*. PhD thesis, University of Pittsburgh, 1997.
6. L. Amgoud and C. Cayrol. On the acceptability of arguments in preference-based argumentation. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 1–7, 1998.

[13] As evidenced by a spate of articles such as the *Role of AI in Law*, published in *The Times* newspaper and available at http://raconteur.net/business/time-for-technology-to-take-over, and the many discussion threads on LinkedIn group for the International Association for AI and Law.

7. M. Araszkiewicz, A. Łopatkiewicz, and A. Zienkiewicz. Parent plan support system–context, functions and knowledge base. In *Business Information Systems Workshops*, pages 160–171. Springer, 2013.
8. M. Araszkiewicz, A. Lopatkiewicz, A. Zienkiewicz, and T. Zurek. Representation of an actual divorce dispute in the parenting plan support system. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law, ICAIL 2015, San Diego, CA, USA, June 8-12, 2015*, pages 166–170.
9. K. Ashley. *Modelling Legal Argument: Reasoning with Cases and Hypotheticals*. Bradford Books/MIT Press, Cambridge, MA, 1990.
10. K. Ashley and S. Brüninghaus. A predictive role for intermediate legal concepts. In *Proceedings of JURIX 2003*, pages 1–10, 2003.
11. K. Atkinson, T. Bench-Capon, H. Prakken, and A. Wyner. Argumentation schemes for reasoning about factors with dimensions. *In Proceedings of JURIX 2013*, pages 39–48, 2013.
12. T. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.
13. T. Bench-Capon. Relating values in a series of supreme court decisions. In *Proceedings of JURIX 2011*, pages 13–22. IOS Press, 2011.
14. T. Bench-Capon and E. Rissland. Back to the future: dimensions revisited. In *Proceedings of JURIX 2001*, pages 41–52. IOS Press, 2001.
15. T. Bench-Capon and G. Sartor. A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence*, 150(1-2):97–143, 2003.
16. T. J. M. Bench-Capon. Representation of case law as an argumentation framework. *Legal Knowledge and Information Systems, Proceedings of Jurix 2002*, pages 103–112, 2002.
17. T. J. M. Bench-Capon. Try to see it my way: Modelling persuasion in legal discourse. *Artif. Intell. Law*, 11(4):271–287, 2003.
18. T. J. M. Bench-Capon. Representing popov v hayashi with dimensions and factors. *Artif. Intell. Law*, 20(1):15–35, 2012.
19. T. J. M. Bench-Capon and F. Coenen. Isomorphism and legal knowledge based systems. *Artif. Intell. Law*, 1(1):65–86.
20. T. J. M. Bench-Capon and S. Modgil. Case law in extended argumentation frameworks. In *The 12th International Conference on Artificial Intelligence and Law, Proceedings of the Conference, June 8-12, 2009, Barcelona, Spain*, pages 118–127. ACM Press, 2009.
21. T. J. M. Bench-Capon and H. Prakken. Using argument schemes for hypothetical reasoning in law. *Artif. Intell. Law*, 18(2):153–174, 2010.
22. T. J. M. Bench-Capon, G. O. Robinson, T. Routen, and M. J. Sergot. Logic programming for large scale applications in law: A formalisation of supplementary benefit legislation. In *Proceedings of the First International Conference on Artificial Intelligence and Law*, pages 190–198, 1987.
23. D. Berman and C. Hafner. Representing teleological structure in case-based legal reasoning: The missing link. In *Proceedings of the Fourth International Conference on Artificial intelligence and Law*, pages 50–59, 1993.
24. D. H. Berman and C. D. Hafner. Understanding precedents in a temporal context of evolving legal doctrine. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 42–51, 1995.
25. F. J. Bex. *Arguments, stories and criminal evidence: A formal hybrid theory*, volume 92. Springer, 2011.

26. K. Branting. Building explanations from rules and structured cases. *International Journal of Man-Machine Studies*, 34(6):797–837, 1991.
27. L. K. Branting. Reasoning with portions of precedents. In *Proceedings of the 3rd international conference on Artificial intelligence and law*, pages 145–154. ACM, 1991.
28. L. K. Branting. An issue-oriented approach to judicial document assembly. In *Proceedings of the 4th international conference on Artificial intelligence and law*, pages 228–235. ACM, 1993.
29. P. Bratley, J. Frémont, E. Mackaay, and D. Poulin. Coping with change. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, pages 69–76, 1991.
30. J. Breuker and N. den Haan. Separating world and regulation knowledge: Where is the logic. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, pages 92–97. ACM Press, 1991.
31. G. Brewka, P. Dunne, and S. Woltran. Relating the semantics of abstract dialectical frameworks and standard afs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 780–785, 2011.
32. G. Brewka, H. Strass, S. Ellmauthaler, J. Wallner, and S. Woltran. Abstract dialectical frameworks revisited. In *23rd International Joint Conference on Artificial Intelligence*, 2013.
33. G. Brewka and S. Woltran. Abstract dialectical frameworks. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference*, 2010.
34. S. Brüninghaus and K. Ashley. Predicting outcomes of case-based legal arguments. In *9th International Conference on Artificial Intelligence and Law*, pages 233–242, 2003.
35. C. Cayrol and M.-C. Lagasquie-Schiex. Bipolar abstract argumentation systems. In *Argumentation in Artificial Intelligence*, pages pp 65–84. Springer US, 2009.
36. A. Chorley and T. Bench-Capon. Agatha: Using heuristic search to automate the construction of case law theories. *Artificial Intelligence and Law*, 13(1):9–51, 2005.
37. A. Chorley and T. Bench-Capon. An empirical investigation of reasoning with legal cases through theory construction and application. *Artif. Intell. Law*, 13(3-4):323–371, 2005.
38. K. L. Clark. Negation as failure. In *Logic and data bases*, pages 293–322. Springer, 1978.
39. T. M. Connolly and C. E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
40. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and *n*-person games. *Artificial Intelligence*, 77:321–357, 1995.
41. S. Ellmauthaler and H. Strass. The DIAMOND system for computing with abstract dialectical frameworks. In *Computational Models of Argument - Proceedings of COMMA 2014, Atholl Palace Hotel, Scottish Highlands, UK, September 9-12, 2014*, pages 233–240, 2014.
42. A. v. d. L. Gardner. *Artificial intelligence approach to legal reasoning*. MIT Press, Cambridge, MA, 1984.
43. M. Grabmair and K. D. Ashley. Argumentation with value judgments - an example of hypothetical reasoning. In *Legal Knowledge and Information Systems - JURIX 2010: The Twenty-Third Annual Conference on Legal Knowledge and Information Systems, Liverpool, UK, 16-17 December 2010*, pages 67–76, 2010.

44. T. R. Gruber. The role of common ontology in achieving sharable, reusable knowledge bases. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 601–602, 1991.
45. C. D. Hafner. Conceptual organization of case law knowledge bases. In *Proceedings of the First International Conference on AI and Law*, pages 35–42, 1987.
46. J. Henderson and T. J. M. Bench-Capon. Dynamic arguments in a case law domain. In *Proceedings of the Eigths International Conference on Artificial Intelligence and Law*, pages 60–69, 2001.
47. J. Horty and T. Bench-Capon. A factor-based definition of precedential constraint. *Artif. Intell. Law*, 20(2):181–214, 2012.
48. J. F. Horty. Reasons and precedent. In *The 13th International Conference on Artificial Intelligence and Law*, pages 41–50, 2011.
49. P. Johnson and D. Mead. Legislative knowledge base systems for public administration: Some practical issues. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, pages 108–117, 1991.
50. P. Johnson and D. Mead. Rule based system and method for writing, developing, implementing and administering legislation, Oct. 23 2007. US Patent 7,287,016.
51. P. R. Johnson, T. J. Reid, and A. Barry. Rule based system and method, Apr. 26 2011. US Patent 7,933,854.
52. R. W. V. Kralingen. *Frame-based conceptual models of statute law*. Kluwer Law Intl, 1995.
53. E. H. Levi. *An introduction to legal reasoning*. University of Chicago Press, 1949.
54. L. Lindahl and J. Odelstad. Intermediaries and intervenients in normative systems. *J. Applied Logic*, 6(2):229–250, 2008.
55. L. T. McCarty. Reflections on taxman: An experiment in artificial intelligence and legal reasoning. *Harvard Law Review*, pages 837–893, 1977.
56. S. Modgil and T. J. M. Bench-Capon. Metalevel argumentation. *J. Log. Comput.*, 21(6):959–1003, 2011.
57. M. Palmirani, G. Governatori, and G. Contissa. Modelling temporal legal rules. In *The 13th International Conference on Artificial Intelligence and Law*, pages 131–135, 2011.
58. H. Prakken. From logic to dialectics in legal argument. In *Proceedings of the 5th International Conference on Artificial Intelligence and Law*, pages 165–174, New York, NY, USA, 1995. ACM.
59. H. Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.
60. H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game. *Artif. Intell. Law*, 6(2-4):231–287, 1998.
61. R. S. Pressman. *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005.
62. E. L. Rissland. Dimension-based analysis of hypotheticals from supreme court oral argument. In *Second International Conference on Artificial Intelligence and Law*, pages 111–120, 1989.
63. T. Routen and T. J. M. Bench-Capon. Hierarchical formalizations. *International Journal of Man-Machine Studies*, 35(1):69–93, 1991.
64. M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The british nationality act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.
65. T. M. van Engers. Power: Using UML/OCL for modelling legislation - an application report. In *Proceedings of the Eighth International Conference on Artificial Intelligence*

*and Law*, pages 157–167, 2001.
66. R. W. Van Kralingen, P. R. Visser, T. J. Bench-Capon, and H. J. Van Den Herik. A principled approach to developing legal knowledge systems. *International Journal of Human-Computer Studies*, 51(6):1127–1154, 1999.
67. P. R. Visser. *Knowledge specification for multiple legal tasks: A case study of the interaction problem in the legal domain*, volume 17. Kluwer Law Intl, 1995.
68. A. Wyner and R. Hoekstra. A legal case OWL ontology with an instantiation of popov v. hayashi. *Artif. Intell. Law*, 20(1):83–107, 2012.
69. A. Z. Wyner, T. J. M. Bench-Capon, and K. Atkinson. Arguments, values and baseballs: Representation of popov v. hayashi. In *Legal Knowledge and Information Systems - JURIX 2007: The Twentieth Annual Conference on Legal Knowledge and Information Systems*, pages 151–160, 2007.
70. T. Zurek and M. Araszkiewicz. Modeling teleological interpretation. In *Proceedings of the 14th International Conference on Artificial Intelligence and Law*, pages 160–168, 2013.