Alison Chorley and Trevor Bench-Capon, 'AGATHA: Automation of the Construction of Theories in Case Law Domains' in T. Gordon (ed.), *Legal Knowledge and Information Systems. Jurix 2004: The Seventeenth Annual Conference*. Amsterdam: IOS Press, 2004, pp. 89-98.

AGATHA: Automation of the Construction of Theories in Case Law Domains

Alison Chorley and Trevor Bench-Capon
Department of Computer Science, The University of Liverpool, Liverpool, UK

Abstract. Some recent accounts of reasoning with legal cases view reasoning with cases as theory construction. In this paper we describe AGATHA (ArGument Agent for THeory Automation) which will automatically generate theories intended to explain a body of case law by following a process inspired by the style of argumentation found in case based reasoning systems. Thus AGATHA behaves like a case based reasoner, but has as its end product a theory, which can be examined, critiqued or input to our other tool, CATE, for refinement or to generate executable code.

1 Introduction

The importance of cases in legal reasoning has been recognised throughout the development of AI and Law. Even approaches which took formalisation of legislation as their starting point, e.g. [16], rapidly came to realise that crucial questions of the interpretation and application of terms found in the legislation could be answered only be reference to cases (e.g. [5]). Cases, despite some differences in the ways in which they are used, are of considerable importance in Civil Law jurisdictions as well as Common Law jurisdictions [3], [14]. Given this centrality of cases, a good understanding of their contribution and use is vital.

Despite the recognition of the importance of cases, there has been less agreement on the way in which cases should be represented and used within AI and Law systems. We may distinguish approaches which have used cases as a knowledge source, (e.g. [4]) on a par with other sources such as statutes and commentaries, and those which have placed importance on the structure and manipulation of cases as entities in their own right, as in, for example, the various systems originating in HYPO [2], [1], [17], [7]. In the first approach cases will be represented only implicitly, whereas in the second they must be represented explicitly. Both these approaches capture aspects of the truth. Given a body of case law, lawyers experienced in the field will be able to give rule like advice: for example we may, with confidence, on the basis of case law, say that injury during a standard commute to work will not be considered as "arising out of, or in the course of, employment" and so not attract Industrial Injury compensation. On the other hand, when it comes to forming an argument in the context of a particular legal case, precedents will be explicitly deployed, in the manner of the HYPO like systems. Both approaches have their strong points: on the first approach we can examine the knowledge that the system will apply: such verifiability may be essential, for example, if we are to trust the operation of a system in administrative law. Moreover we can examine the knowledge to critique the law, identifying areas where we are dissatisfied with it and perhaps propose amendments to legislation accordingly. On the other hand, such systems involve potentially subjective interpretation to extract rules from the case, and do not provide very satisfactory models of legal reasoning. Also they fix the theory, whereas in practice, the interpretation of cases is, at least potentially, continually open to reconsideration (e.g. [12]). The second approach means that each new situation is thought through afresh on its particular merits, rather than being decided mechanically.

A middle way, which attempts to include both aspects, is to introduce the notion of theory construction. For example McCarty states:

"The task for a lawyer or a judge in a "hard case" is to construct a theory of the disputed rules that produces the desired legal result, and then to persuade the relevant audience that this theory is preferable to any theories offered by an opponent" ([13], p285).

On this view, there is a body of knowledge, in the form of a theory, but these theories are always, at least in principle, constructed afresh when a new case appears: thus the theory will be subject to modification in the light of the context provided by difficult cases. We therefore attain the benefits of both approaches: the theory provides the knowledge for inspection (and criticism and modification), and the process of construction can reflect the practice of legal argument.

One recent account of reasoning with legal cases which develops on this view of reasoning with cases as theory construction is the work of Bench-Capon and Sartor, most fully reported in [6]. In this paper we describe AGATHA (ArGument Agent for THeory Automation) which will automatically generate theories as described by [6], by following a process inspired by the style of argumentation found in case based reasoning systems. Thus AGATHA behaves like a case based reasoner, but has as its end product a theory, which can be examined, critiqued or input to our other tool, CATE [10], [11], for refinement or to generate executable code.

2 Argument Moves in HYPO and CATO

When thinking about how to argue a new case on the basis of case law, it appears that people think in terms of analogising a past case to the problem or by distinguishing an unfavourable case, rather than in terms of the theory constructors proposed in [6]. Therefore we wish AGATHA to operate by following a series of argument moves as found in case based reasoners. We therefore take the moves of HYPO [2] and CATO [1] as our starting point.

HYPO creates 3-ply arguments using these four moves:

1) Analogising a problem to a past case with a favourable outcome.
2) Distinguishing a case with unfavourable outcome.
3) Citing a more on point counterexample to a case cited by an opponent.
4) Citing an as-on-point counterexample.

Either party may start the argument by using the first move and analogising the problem case to a past case. The opposing party can then use the remaining three moves to distinguish or counter the cited case. Then the original party can respond completing the 3-ply argument.

CATO extended HYPO with four extra moves:

5) Downplaying the significance of a distinction.
6) Emphasising the significance of a distinction.
7) Citing a favourable case to emphasise strengths.
8) Citing a favourable case to argue that weaknesses are not fatal.

Again the argument is started by one party using the first move to analogise a past case to the problem case. The opponent can then respond to this move using another move and then the original party can respond.

Although HYPO is designed simply to create the arguments whereas CATO is designed to support law students learning how to argue with cases and which move to make at each stage in the argument, CATO can also create its own arguments and explain them. As we will target the basic theory of [6] rather than the extension designed to allow downplaying

and emphasising distinctions, moves 5 and 6 of CATO cannot be used. In any event we would argue that these concern theory evaluation rather than theory construction. Also we will not adopt moves 7 and 8 from CATO at this stage. Arguably these moves also relate to evaluation as they strengthen rather than develop the theory. We therefore base AGATHA on the moves found in HYPO, although we use the factor based representation of cases used in CATO rather than dimensions as found in HYPO: again this is because we are using the basic theory of [6], rather than the extended notion which incorporates dimensions.

AGATHA models the four moves described in HYPO although the distinguish move is expressed as three distinct moves, depending on whether it is the citation of a case, a rule preference or a value preference which is advanced to support the opposing view. The counter example moves have been merged, since AGATHA only uses the most-on-point cases available. Again the distinction between moves 3 and 4 relates more to evaluation that construction. This gives AGATHA five moves, which we describe in the next section. The idea is that AGATHA will use these moves to simulate a dialogue between the plaintiff and the defendant, constructing the theory as a side effect of the dialogue.

3 Argument Moves in AGATHA

Cases are represented as sets of factors and an outcome, which is either plaintiff or defendant. Factors, which represent particular relevant aspects of a case, are represented as a factor name, an outcome favoured by the present of that factor, and a value which is the reason why that factor favours that outcome.

The five moves available in AGATHA are: Analogise Case, Distinguish with Arbitrary Preference, Distinguish with Case, Distinguish Problem, and Counter with Case.

1. *Analogise Case.* This move cites a precedent case which has the outcome the party making the move desires. The factors which are present in both the problem case and the case being cited are sorted into the factors which support that outcome and those factors which support the opposite outcome. A rule preference is made with the supporting factors preferred over the contrary factors. This move follows the method of extracting rules from cases proposed in [15].

The first move made has to be *Analogise Case*. *Analogise Case* can also follow *Distinguish with Arbitrary Preference*. It cannot follow the other three moves.

2. *Distinguish with Case.* This move distinguishes a case already cited in the debate and cites a new case which has the different outcome. To distinguish the previously cited case, AGATHA takes all the factors not used in the Analogise case move which support the outcome and adds them to the factors used in the rule preference from the cited case. So, for example, if the previously cited case was a defendant case, AGATHA takes the unused defendant factors from that case and adds them to the used defendant factors. This creates a larger rule containing all the defendant factors from the case which is then preferred over the original plaintiff factors. This gives a more complex rule which can be used to decide the previously cited case but cannot be used to decide the problem case because this case does not contain all the factors contained in the new rule preference. AGATHA then cites a precedent case with a different outcome from the previously cited case, to give a theory supporting the other side.

Distinguish with Case can follow the *Analogise Case*, *Distinguish with Case* and *Counter with Case* moves because these all cite a new case. It cannot follow the *Distinguish with Arbitrary Preference* and *Distinguish Problem* as these do not cite a case.

3. *Distinguish with Arbitrary Preference*. This move distinguishes the previously cited case in the same way as for the *Distinguish with Case* move, but instead of analogising a new case, AGATHA makes an arbitrary preference using the factors from the problem case that

are included in the theory constructed by the analogising move and only these factors. If, for example, AGATHA is making a plaintiff move, the arbitrary preference has the plaintiff factors preferred over the defendant factors, otherwise, for a defendant move, the defendant factors are preferred over the plaintiff factors. The preference is arbitrary because there is no support for the preference; it just depends on what the party making the move needs to assume to make their case.

It can follow the *Analogise Case, Distinguish with Case* and *Counter with Case* moves because they all cite a new case. It cannot follow the *Distinguish with Arbitrary Preference* and *Distinguish Problem* as these do not cite a new case.

4. *Distinguish Problem.* This move distinguishes the problem case instead of the previously cited case. If, for example, AGATHA is making a plaintiff move, it takes all the plaintiff factors from the problem case and conjoins them as the antecedent into a single rule with plaintiff as consequent.. The defendant factors from the problem case are similarly conjoined as the antecedent of a single rule with defendant as consequent. Next the value sets comprising the values associated with the factors in the two rules are created and a value preference is created with the value set corresponding to the plaintiff factors being preferred over the value set from the defendant factors. Finally a rule preference is created using this value preference.

It can follow the *Analogise Case, Distinguish with Case* and *Counter with Case* moves because they all cite a new case. It cannot follow the *Distinguish with Arbitrary Preference* and *Distinguish Problem* as these do not cite a new case.

5. *Counter with Case.* This move counters the previously cited case by finding a case which is as-on-point or more-on-point as the previous case but was decided for the other side. For an as-on-point counter move, the new case must have the same factors matching the problem case as the previously cited case. The original rule and value preferences which are supported by the previously cited case are replaced with new preferences which are opposite to the original preferences and are supported by the new case.

For a more-on-point counter move, the new case must have the same factors matching the problem case as the previously cited case and extra factors which match the problem case but are not present in the previously cited case. The original rule and value preferences supported by the previously cited case are replaced by new preferences which are supported by the new case.

It can follow the *Analogise Case, Distinguish with Case* and *Counter with Case* moves because they all cite a new case. It cannot follow the *Distinguish with Arbitrary Preference* and *Distinguish Problem* as these do not cite a case.

The moves can only be made once to a given theory apart from *Distinguish with Arbitrary Preference* which can be made more than once, and *Counter with Case* which can be made once for each of the cases which are most-on point. Note also that AGATHA may extend beyond the third ply if moves are available to do so.

The argument moves used in AGATHA use the theory constructors from [6], [8] and [9] to create the underlying theory. When a move is made, a number of theory constructors are applied to extend the current theory. For example, the *Analogise Case* move uses the *Include Case* constructor to include the cited case into the theory, the *Include Factor* constructor to include all the matching factors with the problem case and the *Merge Factors* constructor to merge the plaintiff and defendant factors together. Finally it uses the *Preferences from Case* constructor to include the rule preference which is used to explain the decision for the cited case. Table 1 shows the Theory Constructers which are used in each Move.

Table 1: Table of Theory Constructors used in each move.

Move	**Theory Constructors Used**
Analogise Case	Include Case Include Factor Merge Factors Preferences from Case
Distinguish with Case	Include Case Include Factor Merge Factors Preferences from Case Remove Rule Preference
Distinguish with Arbitrary Preference	Include Factor Merge Factors Remove Rule Preference Preferences from Case Arbitrary Preference
Distinguish Problem	Include Factor Merge Factors Value Preference Rule Preference From Value Preference
Counter with Case	Include Case Include Factor Merge Factors Preferences from Case Remove Rule Preference

4 AGATHA Program

When AGATHA is first started it prompts the user to create a new project or to open an existing project. If the user chooses to start a new project they are asked to choose the problem case which is to be decided and then they are asked to choose the set of cases to be used to explain the decision for the problem case. If the user chooses to open an existing project they can then modify the project by replacing the problem case or by adding or removing cases from the set of precedent cases. When the project has been given a name, AGATHA runs the theory constructor to create all the theories that can be produced in this context.

AGATHA's interface, shown in Figure 1, contains buttons along the top of the interface that, when selected, allow the user to open an existing project, start a new project or modify the currently open project. The user can also choose to execute all the theories in the project to produce a table of the theories and the corresponding outcomes for the problem case.

When AGATHA begins theory construction, it creates an initial theory (Theory 0) which only contains the problem case and places it in the *Working* folder. AGATHA then takes the lowest numbered theory (Theory 0 when the theory construction starts) and applies all the moves which are applicable to it. For Theory 0, AGATHA can apply both plaintiff and defendant moves but for all the subsequent theories the players must alternate, so a plaintiff move must follow a defendant move, and vice versa. AGATHA checks which moves can be made by checking the preconditions for each move against the theory. If the preconditions match it applies the move. Each move that can be applied produces a new theory. When alternative moves are available, a new branch is added to the tree of theories being created.

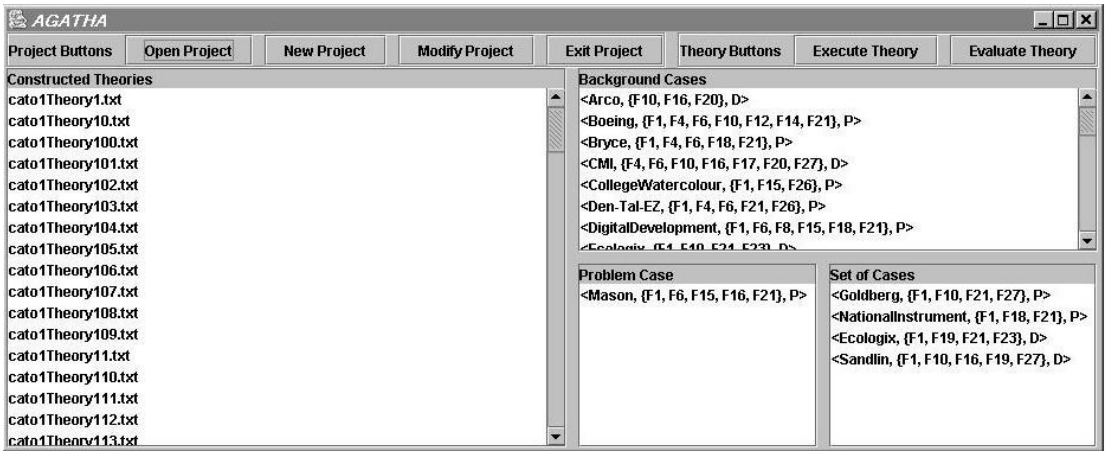

Figure 1: Screen shot of AGATHA

As each move is applied to the theory, the resulting theories are checked and only those which give the same outcome for the problem case as the party making the move are retained. If the move made does not give the correct outcome, the theory is discarded because even though the move could be applied to the theory it has not helped the party making the move, and so does not represent a sensible move. Theories are then saved to the *Working* folder and numbered sequentially.

When all legal moves have been applied to the node, the theory is moved to the *Theory* folder. AGATHA then applies the process to the lowest numbered theory remaining in the *Working* folder and continues until the *Working* folder has no more theories, at which point the project is complete and all the resulting theories are displayed in the *Constructed Theory* window on the interface. The effect of this is to give a breadth first construction of the tree of theories.

5 Wild Animals Example

This illustrative example uses the widely discussed wild animal cases used in [6] and [11]. This small example allows an exhaustive walk through of the operation of AGATHA.

In all three cases, the plaintiff (P) was chasing wild animals, and the defendant (D) interrupted the chase, preventing P from capturing those animals. The issue to be decided is whether or not P has a legal remedy (a right to be compensated for the loss of the game) against D. In the first case, *Pierson v Post*, P was hunting a fox on open land in the traditional manner using horse and hound, when D killed and carried off the fox. In this case P was held to have no right to the fox because he had gained no possession of it. In the second case, *Keeble v Hickeringill*, P owned a pond and made his living by luring wild ducks there with decoys, shooting them, and selling them for food. Out of malice, D used guns to scare the ducks away from the pond. Here P won. In the third case, *Young v Hitchens*, both parties were commercial fisherman. While P was closing his nets, D sped into the gap, spread his own net and caught the fish. In this case D won.

As analysed in [6], the cases can be described using four factors and three values. The factors are: the plaintiff did not have possession of the animal (*pNposs*), the plaintiff owned the land (*pLand*), the plaintiff was pursuing his livelihood (*pLiv*) and the defendant was pursuing his livelihood (*dLiv*). The first is intended to reduce litigation (*LLit*); the second to secure enjoyment of property rights (*MSec*) and the last two to promote productive activity (*MProd*).

Young is taken as the problem case with *Pierson* and *Keeble* as the set of cases that

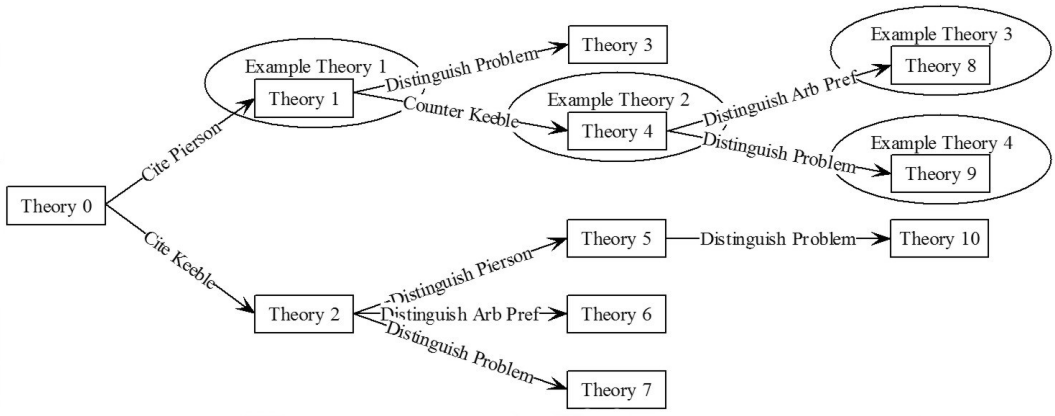

Figure 2: Theories Constructed by AGATHA.

```
Theory Cases :  <Young, {pLiv, pNposs, dLiv}, D>
Theory Factors :
Theory Rules :
Theory Preferences :

Theory Value Preferences :
```

Figure 3: Theory 0.

AGATHA can use to create the theories. *Keeble* is a plaintiff case and has two factor matches with the problem case. *Pierson* is a defendant case and has one factor matching with the problem case.

Using all the moves defined in AGATHA, AGATHA creates ten theories which are shown in Figure 2. Figure 2 also shows how the theories relate to each other. The rules, rule preferences and value preferences for the subsequent theories are shown in Table 2.

From Theory 0 (Figure 3) only *Analogise Case* can be made. First the defendant move is made by analogising *Pierson* to the problem case to produce Theory 1 (Table 2). Then the plaintiff move is made by analogising *Keeble* to the problem case to produce Theory 2.

From Theory 1 the *Distinguish with Case* and *Distinguish with Arbitrary Preference* moves cannot be made because there are no extra factors that can be used to distinguish *Pierson*. *Distinguish Problem* can be made to distinguish *Young* and produce Theory 3. *Counter with Case* can be made because *Keeble* is more-on-point than *Pierson* and produces Theory 4. Although, as discussed below, these theories contain the same rules and preferences, the justification of the rules and preferences and the moves available may be different for each theory.

From Theory 2 *Distinguish with Case* can be used to distinguish Keeble and cite Pierson to produce Theory 5, *Distinguish with Arbitrary Preference* produces Theory 6 and *Distinguish Problem* produces Theory 7. *Counter with Case* cannot be used because Pierson is less-on-point than Keeble

From Theory 3 there are no moves that can be made so this line of the dialogue stops

From Theory 4 *Distinguish with Case* and *Counter with Case* cannot be used because there are no more defendant cases to be cited. *Distinguish with Arbitrary Preference* produces Theory 8 and *Distinguish Problem* produces Theory 9.

From Theory 5 *Distinguish with Case* and *Distinguish with Arbitrary Preference* cannot be used because Pierson has no more factors that could be used to distinguish it. *Distinguish Problem* produces Theory 10. Note that the alternative way of distinguishing the problem, by preferring MSec to LLit cannot be used because pLand is not present in *Young*, and so

Table 2: The rules, rule preferences and value preferences for the theories

Theory	Rules	Rule Preference	Value Preference
1	(1) pNposs →D		
2,3,4	(1) pNposs →D (2) pLiv →P	(2) > (1)	MProd > LLit
5	(1) pNposs →D (2) pLiv →P (3) pLand →P (4) {pLiv, pLand} →P	(4) > (1)	{MProd, MSec} > LLit
6,8	(1) pNposs →D (2) pLiv →P (3) pLand →P (4) {pLiv, pLand} →P	(4) > (1) (1) > (2)	{MProd, MSec} > LLit LLit > MProd
7,9	(1) pNposs →D (2) pLiv →P (3) dLiv →D (4) {dLiv, pNposs} →D	(2) > (1) (4) > (2)	MProd > LLit {LLit, MProd} > MProd
10	(1) pNposs →D (2) pLiv →P (3) pLand →P 4) {pLiv, pLand} →P	(4) > (1) (2) > (1)	{MProd, MSec} > LLit MProd > LLit

this would not produce a pro-plaintiff theory for *Young*. *Counter with Case* cannot be used as there are no more defendant cases that can be used.

From Theory 6 the only potential move is *Analogise Case*, but this move cannot be used because there are no remaining plaintiff cases.

From Theory 7 there are no moves that can be used.

From Theory 8 the only potential move *Analogise Case*, but again this move cannot be used because there are no remaining plaintiff cases.

From Theory 9 there are no moves that can be used.

From Theory 10 there are no moves that can be used.

The tree is therefore complete.

From an analysis of the preference sections of the theories, it can be seen that several theories have identical preferences, even though these preferences may have different labels and have been produced using different moves. There are three groups of identical theories and two theories which are different.

The first group of theories contains Theories 2, 3 and 4. Theory 2 and Theory 4 are identical because *Pierson* only has one factor and so cannot contribute a rule preference so, for Theory 4 when *Counter with Case* is used, *Keeble* contributes the same rule preference as *Analogise Case* for Theory 2. Theory 3 is a plaintiff theory and so takes the defendant *pNposs* factor from Theory 1 and adds the plaintiff factor from the *Young* case description and creates a rule preference of *(pLiv→P > pNposs→D)* which is the same rule preference which Keeble contributes.

The second group contains Theories 6 and 8. These are identical because their preceding theories are also identical (Theory 2 proceeds Theory 6 and Theory 4 precedes Theory 8) and they are produced by making the same move.

The third and final group contains Theories 7 and 9 and they are identical due to the same reasoning as for the second group.

Theory 5 is a distinct theory. To create Theory 5 from Theory 2, *Keeble* is distinguished and *Pierson* is cited but *Pierson* only has a single factor which is already present in the theory and so does not contribute a rule preference. A pro-defendant outcome is produced, however, because the rule preference of *({pLiv, pLand}*$\rightarrow P$*)* over *(pNposs*$\rightarrow D$*)*, is not applicable to *Young*, since pLand is not present, which allows *(pNposs*$\rightarrow D$*)* to fire and give an outcome for the defendant.

6 CATO Example

We have also used AGATHA on out other test domain, US Trade Secrets law, as modelled in [1]. The domain is also described in [8]. This is a larger domain, containing 32 cases, 26 factors and 5 values. Running AGATHA on the case of *Mason versus Jack Daniels* (discussed in [1]), with the limited set of cases of two plaintiff cases and two defendant cases, produces a tree of depth 7 with 114 leaf nodes, of which 29 represent distinct theories. 16 represent an outcome for the plaintiff and 13 represent an outcome for the defendant. Adding a further two cases, gives rise to a theory space with maximum depth of 9 and 996 leaf nodes, representing 202 theories, 112 for the plaintiff and 90 for the defendant.

As the domain becomes larger, the game tree, and hence the theory space, becomes very much larger. This is entirely to be expected, and as this is what invariably happens to a game tree when we move from a simpler to a more complex game. It means, however, that the exhaustive construction of the theory space will not always be the best strategy for realistically large problems, especially if we want to avoid being selective in the inclusion of cases in the background. The response to this is discussed in the next section.

7 Discussion and Future Work

AGATHA has shown that it is possible to construct the space of theories of a case law domain by applying argumentation moves derived from work on reasoning with legal cases. By using these moves AGATHA is following a cognitively plausible strategy, and the sequence of moves is available to present the case to an opponent.

AGATHA currently generates the complete theory space. This space may include duplicate theories, and theories which seem less acceptable than others. In the current version we have made no attempt to prune the space since it may be of interest to see that theories can be reached by different routes, so that it can be seen whether the same solution is independent of who makes the first move, and the order in which moves are made. Moreover different routes may supply different justifications for various theory elements. None the less, as the size of the domain increases, there may be advantages in pruning the space. One obvious way to do this is expand only one instance of identical theories. An alternative, and more promising approach is to exploit the fact that we are dealing with a process that is a two player adversarial game. In such games, it is typically the case that it is impractical to generate the whole game tree and so techniques have been developed to address this problem in the analysis of two player games, using heuristic search techniques.

Therefore a second way of controlling the expansion of the tree is to provide some heuristic to select the moves to apply. We could rank moves according to their potential strength: one plausible ranking would be *Counter with Case, Distinguish with Case, Distinguish Problem, Distinguish with Arbitrary Preference*. In this way only a single branch of the tree would be produced from each node. A second source of expansion is the choice of cases to cite: again some heuristic in term of similarity to the problem case would provide a sensible means of limiting this growth.

A more sophisticated approach would be to use a variety of heuristic search. To do this we will need to have a means of evaluating the theories produced as the tree is developed. We will then be able apply a standard technique to prune the game tree, such $\alpha\beta$ or A* search e.g. [18]. Using such a heuristic search technique will enable the system to produce the theory with results from "best play" by both sides.

The next step therefore is to develop a means to evaluate the theories. To complete our family of programs, we will next develop ETHEL (Evaluation of THEories in Law), which will evaluate theories using the criteria such as those proposed in [6], and including explanatory power, simplicity, freedom from arbitrary preferences and the ability to generalise. When this tool is available, we will use it to add heuristic search to AGATHA.

In this paper we have described AGATHA, a tool which automatically produces the theories which can be constructed to explain a set of cases described using factors. In producing its theories, it follows a strategy derived from case based approaches to reasoning with cases, which can be used to explain and justify the theories produced.

References

[1] Aleven, V. (1997). *Teaching Case Based Argumentation Through an Example and Models.* PhD Thesis. The University of Pittsburgh.

[2] Ashley. K.D., (1990). Modelling Legal Argument. Bradford Books, MIT Press, Cambridge, Mass.

[3] Ashley, K.D. (2004). *Case-Based Models of Legal Reasoning in a Civil Law Context* . Invited paper. International Congress of Comparative Cultures and Legal Systems of the Instituto de Investigaciones Jurídicas, Universidad Nacional Autonoma de México, Mexico City.

[4] Bench-Capon, T., *Practical Legal Expert Systems: the Relation Between a Formalisation of Law and Expert Knowledge.* In Bennun and Narayanan (eds) Computers, Law and AI, Ablex, 1991, pp191-201.

[5] Bench-Capon, T., *Knowledge Based Systems Applied To Law: A Framework for Discussi* on, in T. Bench-Capon (ed), *Knowledge Based Systems and Legal Applications,* Academic Press, 1991, pp329-342.

[6] Bench-Capon, T., & Sartor, G. 2003. A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence* . Vol 150 1-2 pp97-143.

[7] Brüninghaus, S. & Ashley, K. D.,. Predicting Outcomes of Case-based Legal Arguments. Proceedings of the Ninth International Conference on AI and Law: ACM Press, New York, 2002, pp233-42.

[8] Chorley, A. & Bench-Capon, T., Developing Legal Knowledge Based Systems Through Theory Construction. Technical Report ULCS-03-013, Department of Computer Science, The University of Liverpool, 2003. http://www.csc.liv.ac.uk/research/techreports/tr2003/ulcs-03-013.pdf

[9] Chorley, A. & Bench-Capon, T., Developing Legal Knowledge Based Systems Through Theory Construction. Proceedings of the Ninth International Conference on AI and Law: ACM Press, New York, 2002, pp85-6.

[10] Chorley, A. & Bench-Capon, T., (2003). *Reasoning with Legal Cases as Theory Construction: Some Experimental Results* . In D. Bourcier (ed), Proceedings of Jurix 2003. IOS Press Amsterdam. pp173-82

[11] Chorley, A. & Bench-Capon, T., (2004). *Support for Constructing Theories in Case Law Domains* . In F. Galindo, M. Takizawa and R. Traunmuller (eds) Proceedings of DEXA 2004. LNCS 3180. Springer Verlag: Berlin. pp508-517.

[12] Levi, E.H. 1949. *An Introduction to Legal Reasoning* . University of Chicago Press: Chicago.

[13] McCarty, L.T. 1995. *An Implementation of Eisner v Macomber.* In Proceedings of the Fifth International Conference on AI and Law, 276-286. ACM Press: New York.

[14] MacCormick, D.N. &. Summers, R. S. (ed.) 1997. *Interpreting Precedents: A Comparative Study* Dartmouth Publishing: Aldershot, U.K.

[15] Prakken, H. & G. Sartor. 1998. Modelling Reasoning with Precedents in a Formal Dialogue Game. *Artificial Intelligence and Law* 6: 231-287.

[16] Sergot, M., Sadri, F., Kowalski, R., Kriwaczek, F., Hammond, P., Cory, H.: The British Nationality Act as a Logic Program. Commun. ACM 29(5): 370-386 (1986)

[17] Skalak, D.B. & Rissland, E.L. (1992), Arguments and cases: an inevitable intertwining. *Artificial Intelligence and Law* 1: 3-44.

[18] Winston, P.H., (1992). Artificial Intelligence. Addison Wesley: Reading, Mass.