From: ADAPTIVE CONTROL OF ILL-DEFINED SYSTEMS
Edited by Oliver G. Selfridge, Edwina L Rissland,
and Michael A. Arbib
(Plenum Publishing Corporation, 1984)

EXAMPLES AND LEARNING SYSTEMS*

Edwina L. Rissland

Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003, U.S.A.

INTRODUCTION

Any system that learns or adapts -- whether well- or ill-defined, man or machine -- must have examples, experiences, and data on which to base its learning or adaptation. Too often, however, the examples that form the basis of learning are taken for granted. This paper will concentrate on the examples as a study in their own right.

BACKGROUND

The importance of examples to learning systems can be seen in many well-known A.I. programs. For instance, Winston's program [1975] learns the concept of "arch" from a sequence of examples: of arches and non-arches. The initial example, and examples that fail to be an arch in just one aspect, "near misses", are what drive this learning system. Samuel's Checker Player is another classic AI program that makes use of examples [1963, 1967]. Samuel gave his program libraries of specific book moves. His work is an example of the interplay between examples, adaptation and learning systems.

Selfridge's recent COUNT and sorting programs [Selfridge 1979, 1980], as well as his classic PANDEMONIUM [Selfridge 1958], also depend heavily on example data and problems. COUNT would not learn

*The research reported in this paper was supported in part by the National Science Foundation under grant no. IST-8017343.

its task -- how to count the number of symbols in a string -- if not presented with a sequence of challenging but not too difficult problems to solve. The sorting program would not be able to adaptively tune itself -- learn the "best" choice of sorting method to use-- if it did not have abundant experience with problems of sorting lists.

These same observations can also be made about Soloway's BASEBALL program [Soloway 1978], which learns the concepts and rules of baseball by "watching" baseball games; and about Lenat's AM program [Lenat 1977], which discovers new mathematical concepts partly on the basis of how the concepts and control heuristics fare on examples.

Examples also play a critical role in non-computer systems like law, mathematics and linguistics. For instance, the law -- especially common law, which is based on the doctrine of precedent ("stare decisis") -- is a wonderful example of a system which is hard to define (despite its superficial resemblance to a rule-based system) and which runs entirely in response to the examples -- i.e., cases -- presented to it. Cases must be adjudicated; the court expresses the results as "holdings" and "dicta" (rules and comments of the case); these lead to the further evolution of concepts and rules which are then modified by further cases [Levi 1949, Berman 1968]. In legal scholarship and education, the cases may be hypothetical; in some law classes, "hypos" become as "real" and have as much import as actually litigated cases.

Examples are central to reasoning in mathematics in the cycle of "proofs and refutations" [Lakatos 1976]: a mathematical concept is refined in response to how the system, that is the mathematical theory with all of its definitions, theorems, proofs, examples, etc., fares in the face of examples, ranging from standard examples to unusual "monsters". Of course, a rich set of examples is needed for the guessing and inductive inference needed to get the system started in the first place [Polya 1968].

In linguistics, examples too are central to theory formation and in fact, many theories are built in response to what have now become famous examples, or counter-examples, to other theories. (See for instance, Gazdar's [1981, 1982] response to Chomsky's assertion that English does not have phrase-structure grammar.) In fact, Kuhn has pointed out that this interplay between an evolving system and examples is ubiquitous in the history of science [Kuhn 1970].

Finally, in computer programming itself there is an "inevitable intertwining" [Swartout and Balzer 1982] of the evolving system (e.g., a program), as expressed in code and specifications, and the examples used to test it out. One writes code, tests it out on

example data, and then (usually) revises the program; specifications also evolve in this way. The programming and debugging process is thus completely analogous to the cycle of proofs and refutations in mathematics.

Thus, in summary, examples (by which is meant experiences, data, instances) are critical grist for the mill of learning and adaptation. Once this obvious, but key, point is recognized, one is led immediately to questions like the following:

1. How should examples be grouped into types?
2. Do examples have structure?
3. How are examples related to one another?
4. What properties should an example satisfy?
5. Where do the desired properties come from?
6. How can knowledge of examples be applied to ill-defined systems?

The rest of this paper addresses these questions.

TAXONOMIES OF EXAMPLES

When one considers the different effects and uses examples can have with respect to learning systems, one can distinguish different classes of examples. In this section, we briefly review the "epistemological" classes of examples that we have found useful in disciplines like mathematics and law [Rissland 1978, 1982].

It is important to recognize that not all examples serve the same function in learning. For instance, expert teachers and learners know that certain perspicuous ("start-up") examples provide easy access to a new topic, that some ("reference") examples are quite standard and make good illustrations, and that some examples are anomalous and don't seem to fit into one's understanding. We can develop a taxonomy of items based upon how we use them to learn, understand and teach:

(a) start-up examples: perspicuous, easily understood and easily presented cases;
(b) reference examples: standard, ubiquitous cases;
(c) counter examples: limiting, falsifying cases;
(d) model examples: general, paradigmatic cases;
(e) anomalous examples: exceptions and pathological cases.

Start-up examples are simple, easy to understand and explain cases. They are particularly useful when one is learning or explaining a domain for the first time. Such examples can be generated with minimal reference to other examples; thus one can

say they are structurally uncomplicated. A start-up example is often "projective" in the sense that it is indicative of the general case and that what one learns about it can be "lifted" to more complex examples.

Reference examples are examples that one refers to over and over again. They are "textbook cases" which are widely applicable throughout a domain and thus provide a common point of reference through which many concepts, results and other items in the domain are (indirectly) linked together.

Counter-examples are examples that refute or limit. They are typically used to sharpen distinctions between concepts and to refine theorems or conjectures. They are essential to the process of "proofs and refutations" [Lakatos 1976].

Model examples are examples that are paradigmatic and generic. They suggest and summarize expectations and default assumptions about the general case. Thus, they are like "templates" or "frames" [Minsky 1975].

Anomalous examples are examples that do not seem to fit into one's knowledge of the domain, and yet they seem important. They are "funny" cases that nag at one's understanding. Sometimes resolving where they fit leads to a new level of understanding.

An example of applying this classification scheme for an introductory study of continuity from the domain of real function theory might classify: the function f(x)=x as a start-up example; f(x)=x**2, f(x)=e**x as reference examples; f(x)=1/x as a counter-example; "f(x) with no gaps or breaks" as a model example; and f(x)= sin(1/x) as an anomalous example. The first example, f(x)=x, is also a reference example (the "identity" function). Thus, such a classification need not be exclusive. The anomaly sin(1/x) will most likely become a favorite counter-example as one understands that a function can fail to be continuous in at least two ways, that is, by having gaps and breaks and by failing to settle down to a limit. Thus, such a classification is not static. Increased understanding will of course lead to qualifications on the above model of a continuous function, although it will still serve to summarize one's expectations.

In introductory LISP programming one deals with lists of atoms. For the novice, the lists (A) and (A B C) are start-up examples; (A B C) is also a reference example; NIL or ( ), a counter-example; "left-paren atom atom atom ... right-paren", a model example.

The point here is not the particular parsing out of examples into classes, for this depends on many considerations such as one's

level of expertise and purposes for taxonomizing; but that there are classes of examples, and that such epistemological knowledge about classifying examples (and other items) is an important part of one's understanding. Such knowledge can be used to help focus one's attention in learning and explaining, for instance by suggesting heuristics like "Check out the conjecture on reference examples before believing too strongly in it" or "Look for counter-examples in the class of known counter-examples".

STRUCTURAL ASPECTS OF EXAMPLES

In a complex domain like mathematics or law, there are several types of structure. We can distinguish items, relations, spaces:

(1) items are strongly bound clusters of information: for instance, the statement of a theorem, its name, its proof, a diagram, an evaluation of its importance, and remarks on its limitations and generality.

(2) relations between items: for instance, the logical connections between results, such as predecessor results on which a result depends logically and successor results which depend on it.

(3) spaces are sets of similar types of items related in similar ways: for instance, proved results and their logical dependencies. Such a set of items and relations constitute a space, in the case of results, a Results-space.

In essence the idea is that examples (and other items) are cohesive clusters of information which do not exist in isolation from one another; there are relations between them. What distinguishes a "space" from a "set" is the prominence of the relations. The structure of a complex domain like mathematics contains not just one but many spaces, each of which describes a different aspect of knowledge. Examples are but one type of "item" that comprise the knowledge in such domains; others include concepts, results, strategies and goals [Rissland 1978, 1981b . Of concern in this paper are examples by which we mean specific situations, data or experiences.

An example has many aspects or pieces of information that comprise it: its name, taggings and annotations as to epistemological class and importance, lists of pointers to other examples from which it is constructed and to whose construction it contributes, the process of how it is constructed, a schematic or diagram, pointers to items like definitions in other spaces,

statements of what the example is good for or how it can be misleading, sources of further information about the example.

Examples can be related by constructional derivation of how one example is built from others. Examples plus this relation constitute an Examples-space. For instance, in the LISP programming example, the examples ((A) B C) and (A (B C)) can be thought of as constructionally derived from the reference example list (A B C) by the addition of parentheses.

The construction of a new example from others is a process that is found in many fields, for instance law and mathematics. In teaching the law, one frequently makes use of "hypothetical" examples ("hypos") which are often constructed by modification of a well-known reference example, e.g., a textbook case. In a Socratic discussion between a law professor and his class, the teacher may spin out a sequence of hypos to test out the class's understanding and biases on a doctrinal proposition. Such a sequence might contain increasingly more complex or extreme cases.

The following hypothetical examples are taken from a class discussion in contract law. They were used to point out the difference between the doctrines of "consideration" (which emphasizes what the promisee gives the promisor in return for the promise) and "reliance" (which emphasizes how the promisee acts in reliance on the promise). These are two different ways to approach the problem of determining which contracts are legally enforceable as opposed to which are gratuitous gifts. The base case from which the hypos are constructed through modifications is Dougherty v. Salt, a standard case in a course in contract law [Fuller and Eisenberg 1981].

Hypo1:
Facts: Aunt Tillie says, "Charlie, you are such a nice boy; I promise to give you $10,000."

Hypo2:
Facts: Same as Hypo1 with the addition that Charlie says, "Dear Aunt Tillie, I can't take something for nothing, let me give you my third-grade painting."

Hypo3:
Facts: Same as Hypo2 except that Charlie offers to mow Tillie's lawn.

Hypo4:
Facts: Same as Hypo2 except that Charlie's last name is Picasso.

Hypo5:
Facts: Same as Hypo1 with the addition that Aunt Tillie's assets are in ruin and that keeping her promise to Nephew Charlie means her own children starve.

Hypo6:
Facts: Same as Hypo1 with the addition that Charlie then makes an unreturnable deposit on a new car.

Thus in summary there are internal and external structural aspects to examples. The internal structure of an example concerns the cluster of strongly bound information that comprises the example, including pointers to other items like examples. The external aspects concern the relations among examples, for instance how one example is constructed from others.

CONSTRAINTS AND EXAMPLES

An important aspect of examples with respect to learning systems is the obvious fact that examples possess certain properties. For instance, the "near misses" in Winston's work are examples that fail to be arches in exactly one of the required properties of archness. What perhaps may not be so obvious is that in selecting examples to give to a learning system, one does not pick them at random: examples are generated for a purpose -- like giving evidence for or against a conjecture -- and thus examples are usually (carefully) chosen to possess certain desired properties, which we call constraints.

We have called this process of generating examples that meet prescribed constraints "Constrained Example Generation" or "CEG". In past work, we have described, built, and experimented with a model of the CEG process. See for instance, [Rissland 1980, 1981a, Rissland and Soloway 1980a, 1980b]. It is based upon observations of humans working problems in which they are asked to generate examples satisfying certain constraints. Our model of CEG incorporates three major phases: RETRIEVAL, MODIFICATION, and CONSTRUCTION.

When an example is sought, one can search through one's storehouse of examples for one that matches the properties desired. If one is found, the example generation problem has been solved through RETRIEVAL. In retrieval, there are many semantic and contextual factors -- like the last generated example -- and therefore one is not merely plunging one's hand into an unorganized knowledge base. Thus even though retrieval sounds simple, it can be very complex.

However, when a match is not found, how does one proceed? In many cases, one tries to MODIFY an existing example that is judged to be close to the desired example, or to have the potential for being modified to meet the constraints. Often the order of examples selected for modification is based on judgements of closeness between properties of known examples and the desiderata, that is, how "near" the examples are to what is sought.

If attempts at generation through modification fail, experienced example generators, like teachers or researchers, do not give up; rather they switch to another mode of example generation, which we call CONSTRUCTION. Under construction, we include processes such as combining two simple examples to form a more complex one and instantiation of general model examples or templates to create an instance. Construction is usually more difficult than either retrieval or modification.

General Skeleton of the CEG Model

CEG has subprocesses for: Retrieval, Modification, Construction, Judgement, Control

Presented with a task of generating an example that meets specified constraints, one:

1. SEARCHES for and (possibly) RETRIEVES examples JUDGED to satisfy the constraints from an EXAMPLES KNOWLEDGE BASE (EKB); or

2. MODIFIES existing examples JUDGED to be close to, or having the potential for, fulfilling the constraints with domain-specific MODIFICATION OPERATORS; or

3. CONSTRUCTS an example from domain-specific knowledge, such as definitions, general model examples, principles and more elementary examples.

In examining human protocols, one sees two types of generation: (1) retrieval plus modification; and (2) construction. That is, one does not necessarily try first retrieval, then modification, then construction; sometimes construction is attempted straightaway. Clearly, this model needs many other features to describe the CEG process in its entirety; more details can be found in [Rissland 1981a].

To give the reader an idea of the richness and complexity of

the CEG process, we present here a synopsis of a CEG problem taken from the domain of elementary function theory. The problem is:

> Give an example of a continuous, non-negative function, defined on all the real numbers such that it has the value 1000 at the point x=1 and that the area under its curve is less than 1/1000.

Most protocols for this question began with the subject selecting a function (usually, a familiar reference example function) and then modifying it to bring in into agreement with the specifications of the problem. There were several clusters of responses according to the initial function selected and the stream of the modifications pursued. A typical protocol went as follows [Rissland 1980]:

> "Start with the function for a "normal distribution". Move it to the right so that it is centered over x=1. Now make it "skinny" by squeezing in the sides and stretching the top so that it hits the point (1, 1000)."
>
> "I can make the area as small as I please by squeezing in the sides and feathering off the sides. But to demonstrate that the area is indeed less than 1/1000, I'll have to do an integration, which is going to be a bother."
>
> "Hmmm. My candidate function is smoother than it need be: the problem asked only for continuity and not differentiability. So let me relax my example to be a "hat" function because I know how to find the areas of triangles. That is, make my function be a function with apex at (1, 1000) and with steeply sloping sides down to the x-axis a little bit on either side of of x=1, and 0 outside to the right and left. (This is OK, because you only asked for non-negative.) Again by squeezing, I can make the area under the function (i.e., the triangle's area) be as small as I please, and I'm done."

Notice the important use of such modification operations as "squeezing", "stretching" and "feathering", which are usually not included in the mathematical kit-bag since they lack formality, and descriptors such as "hat" and "apex". All subjects made heavy use of curve sketches and diagrams, and some used their hands to "kinesthetically" describe their functions. Thus the representations and techniques used are very rich.

Another thing observed in all the protocols (of which there were about two dozen for this problem) is that subjects make implicit assumptions -- i.e., impose additional constraints -- about the symmetry of the function (i.e., about the line x=1) and its maximum (i.e., occurring at x=1 and being equal to 1000). There are no specifications about either of these properties in the problem

statement. These are the sort of tacit assumptions that Lakatos [1976] talks about; teasing them out is important to studying both mathematics and cognition.

CONSTRAINT GENERATION

Generating examples from constraints presupposes that there are constraints. That is, that properties of examples can be expressed in a language of constraints, and that one actually knows what one wants in the example sought, that is, the constraints. To put it another way, there is a prior problem of constraint generation.

The constraints are often generated from consideration of one's intended use for the example-to-be. For instance, if one were testing a program known to work on simple cases, one would want examples that are more complex or rare, for instance an anomalous or counter-example. If one could not find a satisfying example in one's "Examples Knowledge Base", perhaps organized as an Examples-space, one would need to generate it. To do this one could express the desiderata for the example in terms of constraints and then proceed with the CEG process.

In hypos for a Socratic discussion in law, such constraints would include pedagogical, rhetorical, doctrinal constraints, such as, those in our previous example, arising from the doctrine of consideration. The constraint upon the object given by Nephew Charlie to his Aunt Tille might be loosely described as "being something of value"; this constraint is then varied from something of little value (the typical third-grade painting), to something of some value (mowing the lawn), to something of great value (a "Picasso" third-grade painting).

If one were giving examples of lists to a neophyte LISP programmer, one would make use of domain-specific constraints having to do with "length" (of the list), "order" (of atoms in the list), "depth" (of certain atoms), and whether the list is a "LAT" (i.e., a list of atoms) or a more complex list [Rissland and Soloway 1980b]. For instance, one might want a list such that: (1) it has length 3, and (2) the depth of the first atom is 3. The list (((A)) B C) satisfies both constraints and may be thought of as generated from the reference example (A B C) by a sequence of modifications affecting depth. The lists (((A) B) C) and (((A B C))), which are also generated by modifications affecting depth through the addition of parentheses, satisfy the second (depth) but not the first (length) constraint. Thus, modifications designed to remedy one constraint deficiency might, in the language of Sussman [1973], "clobber a brother goal", that is, another constraint. Such interactions can make the CEG problem quite complex.

APPLICATIONS TO ILL-DEFINED SYSTEMS

In this section we suggest how examples can be used in probing, debugging, specifying, or otherwise dealing with an ill-defined or not well-understood system.

Examples can be useful for probing a system. Consider the following scenario. We've just logged on and are trying to learn how to use or test a system, or a new feature of it. Suppose it were a program to sort letters into alphabetical order.

To probe this program, we might start off by saying, "If this program really works, it ought to do simple little things; it certainly ought to handle A, B, C. If it fails on A, B, C, we know there's a problem from the very beginning." Suppose we find it's OK on A, B, C, a standard start-up or reference example in this mini-domain.

Now we'd try something a bit more complex, like a longer list, say more of the beginning of the alphabet. This is a slight embellishment of the beginning example, arising from a length constraint. Suppose it works on such an example -- which is good since the program didn't have to do "anything" -- let's give the program an opportunity to exercise itself on some other simple cases like C, B, A or the alphabet in reverse order. Then, we might introduce a couple of letter interchanges like A, C, B or M, O, N, P, R, Q. These involve order as well as length constraints and can be generated with an "interchange" modification.

Thus we're probing the system with a sequence of increasingly complex cases derived from standard simple ones. With a minimal level of confidence in the program established, we could go on to test it on more difficult or limiting cases.

We might now check if the program can handle known troublesome examples like a singleton list, like "A", or the empty list, which are known from experience to be some of the cases that make programs cough and sputter, that is, counter-examples. The singleton list often causes problems because of the false assumption that there's (always) going to be more than one element. A LISP programmer would know of another well-known trouble maker: the empty list, NIL. It is an important case in recursive procedures. In fact, it is a favorite (i.e., reference) example with any LISP hacker. In the sorting domain, there would be some other specific things that are known to cause problems, for instance, repeated elements. Thus we are using knowledge about the examples and about the context and task in evaluating a system.

Note that we are not examining or altering the code, which can be considered a lower level representation of the program; rather we're staying on a representational level more akin to the purposes of the program, that is, what the program is supposed to do, or more realistically, what we believe it's supposed to do. We're dealing with the system at arm's length and just probing and asking if the program appears to work by our experience with it on selected, and well-chosen, examples.

Such probing of a system in this manner is similar to testing theorems. (Again, we have the analogy with mathematics, which has been discussed by De Millo et al. [1980].) One can never show that it works by just trying examples, we can only show, perhaps, that it doesn't work, just as one counter-example can refute a conjecture. However, having a variety of instances that work establishes confidence, even if it doesn't end the verification process. By selecting enough well-chosen cases, one can "span" the possibilities and obtain a sense of the program's correctness. One of the differences between a novice and an expert programmer (or mathematician) seems to be the richness of the "spanning" examples used: novices tend to forget to check the program on complicated or known counter-examples and anomalies, even simple ones like NIL. Thus part of the art of expert programming is epistemological knowledge of the type we have described in previous sections of the paper.

Also note that in this scenario the problem of evaluating the answer required little work on our part: alphabetical order is obvious by inspection. In other cases of generating test data, there's a lot more work involved to being a critic. (See [Dietterich and Buchanan] in this volume.)

These remarks also apply if we are writing or debugging a program. Using specific examples to work out a solution helps one to deal with the complexity or lack of specification of the solution. It might well be the case that the kind of examples used in probing might be different from those in design and implementation.

Since it is impossible to completely specify a program under every condition -- because, for instance, the context of the system is changing or one is not sure of what one wants -- using examples to show what the program should do in certain situations, especially those that matter to the specifier or that are too difficult to describe in symbols or words -- provides another means of describing the program. Together traditional specifications in words, logic or symbols joined with example cases provides a better specification than either alone; each mode compensates for and complements the other.

CONCLUSIONS

In this paper we have discussed examples in their relation to learning systems and in their own right. In particular, we have described the structural aspects of examples, including their internal structure as a strongly bound cluster of information and their external relations to other examples through construction. We have used constraints to approach the problem of generating examples with specific properties and have described the prior problem of constraint generation which involves interactions between the system and the examples or example user. Lastly, we have suggested that examples are central to probing, debugging and even specifying systems: they can probe a system and they can help describe it by showing how it does or should operate.

Thus, in our approach we concentrate on the role that examples (that is, experiences, instances, data) play in systems, well- or ill-defined, and find that they are a rich study in themselves. Not only are they interesting, in fact they are central. As that great polymath Oliver Wendell Holmes put it:

> "The life of the law has not been logic; it has been experience"

And it is experience we are capturing with examples.

REFERENCES

Berman, H. J., "Legal Reasoning". In International Encyclopedia of the Social Sciences, MacMillan, 1968.

De Millo, R.A., Lipton, R.J., Perlis, A.J., "Social Processes and Proofs of Theorems and Programs", Communications of the ACM, Vol. 22, No. 5, May 1979. Reprinted in the Mathematical Intelligencer, Vol. 3, No. 1, 1980.

Fuller, L. L., and M. A. Eisenberg, Basic Contract Law. West Publishing Co., Minn., 1981.

Gazdar, G., "Phrase Structure Grammar". In Jacobson, P., and G.K. Pullun (Eds.) The Nature of Syntactic Representation, Reidel, 1982, pp. 131-187.

Gazdar, G., "Unbounded Dependencies annd Coordinate Structure". Linguistic Inquiry 12 (1981), 155-184.

Kuhn, T. S., The Structure of Scientific Revolutions. Second Edition. University of Chicago Press, 1970.

Lakatos, I., Proofs and Refutations. Cambridge University Press, London, 1976. Also in British Journal for the Philosophy of Science, Vol. 14 , No. 53, May 1963.

Lenat, D. B., "Automatic Theory Formation in Mathematics", Proceedings of Fifth International Joint Conference on Artificial Inteligence. Cambridge, MA., 1977.

Levi, E. H., An Introduction to Legal Reasoning. University of Chicago Press, 1949.

Minsky, M. L., "A Framework for Representing Knowledge". In The Pyschology of Computer Vision, Winston (ed), McGraw-Hill, 1975.

Polya, G., Mathematics and Plausible Reasoning, Volumes I and II. Princeton University Press, 1968.

Restatement, Second, Contracts. American Legal Institute, Philadelphia, 1981.

Rissland, E. L., "Examples in the Legal Domain: Hypotheticals in Contract Law". In Proceedings Fourth Annual Conference of the Cognitive Science Society. Ann Arbor, August 1982.

_____, Constrained Example Generation. COINS Technical Report 81-24, University of Massachusetts, 1981a.

_____, The Structure of Knowledge in Complex Domains. COINS Technical Report 80-07, University of Massachusetts, 1981b. Also to appear in Proceedings NIE-LRDC Conference on Thinking and Learning Skills, Lawrence Erlbaum Associates.

_____, "Example Generation". In Proceedings Third National Conference of the Canadian Society for Computational Studies of Intelligence, Victoria, B. C., May 1980.

_____, The Structure of Mathematical Knowledge. A.I. Tech Report 472, A. I. Laboratory, Massachusetts Institute of Technology, 1978.

_____, "Understanding Understanding Mathematics". Cognitive Science, Vol. 2, No. 4, 1978.

Rissland, E. L., and E. M. Soloway, "Overview of an Example Generation System". In Proc. First National Conference on Artificial Intelligence. Stanford, August 1980a.

_____, "Generating Examples in LISP". Proceedings International Workshop on Program Construction. Bonas, France, 1980b.

Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers". In Computers and Thought, Feigenbaum and Feldman (Eds.), McGraw-Hill, 1963.

_____, "Some Studies in Machine Learning Using the Game of Checkers. II--Recent Progress". In IBM J. Research and Development, 11:601-617, 1967.

Selfridge, O. G., "Pandemonium -- A Paradigm for Learning". In Symposium on the Mechanization of Thought Processes. Teddington, England, 1958.

_____, "An Adaptive Sorting Algorithm". In Proceedings Third National Conference of the Canadian Society for Computational Studies of Intelligence, Victoria, B.C., May 1980.

_____, "COUNT — How a Computer Might Do It". Internal Report, Bolt Beranek and Newman Inc, 1979.

Soloway, E. M., "Learning = Interpretation + Generalization: A Case Study in Knowledge-Directed Learning". COINS Technical Report 78-13, University of Massachusetts, 1978.

Sussman, G. K., A Computational Model of Skill Acquisition. A.I. Tech Report 297, A. I. Laboratory, Massachusetts Institute of Technology, 1973. (Doctoral dissertation.)

Swartout, W. and R. Balzer, "On the Inevitable Intertwining of Specification and Programming". CACM Vol. 25, No. 7, July 1982.

Winston, P. H., "Learning Structural Descriptions from Examples" in The Psychology of Computer Vision, Winston (ed), McGraw-Hill, 1975.

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Artificial Intelligence 150 (2003) 1–15

Artificial Intelligence

www.elsevier.com/locate/artint

AI and Law: A fruitful synergy

Edwina L. Rissland [a,*], Kevin D. Ashley [b], R.P. Loui [c]

[a] *Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA 01003, USA*
[b] *Graduate Program in Intelligent Systems and School of Law, University of Pittsburgh, Pittsburgh, PA 15260, USA*
[c] *Department of Computer Science, Washington University, St. Louis, MO 63130, USA*

Received 29 July 2003

AI and Law is a classic field for AI research: it poses difficult and interesting problems for AI, and its projects inform both AI and its focal domain, the law itself. This special issue reports on a range of projects, from those where the law motivates fundamental research and whose results reach beyond the legal domain, to those that partake of the benefits of techniques and wisdom from AI as a whole. For instance, projects tackling legal argument have not only created programs that produce legal arguments but also led to insights and advances in the logic of argumentation. Projects with an applications bent have often provided insights about the limitations and nuances of existing techniques, and have served as the catalysts for developing new approaches. For instance, harnessing models of legal argument to teach law students how to argue has led to refinements to and extensions of the models. There is a synergy not only between law and AI, but also between AI and AI and Law. In fact, the work on Case-Based Reasoning (CBR) done in the AI and Law community provided one of the most important streams of results that contributed to the birth of that area in the mid-1980s. Currently, work on legal argumentation is having a similar impact on the international non-standard logic and argumentation communities.

AI and Law is much more than an applications area. Its concerns touch upon issues at the very heart of AI: reasoning, representation, and learning. For the AI researcher interested in symbolic methods—or methods of whatever stripe—that are focused on providing explanations and justifications, AI and Law is an excellent arena. No matter how a reasoner arrives at a legal answer it must be explained, justified, compared to and contrasted with alternatives. For the researcher interested in topics like negotiation, decision-making, e-commerce, natural language, information retrieval and extraction, and data mining, AI and Law is a rich source of problems and inspiration.

* Corresponding author.
E-mail addresses: rissland@cs.umass.edu (E.L. Rissland), ashley@pitt.edu (K.D. Ashley), loui@cs.wustl.edu (R.P. Loui).

0004-3702/$ – see front matter
doi:10.1016/S0004-3702(03)00122-X

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Artificial Intelligence 150 (2003) 1–15
Artificial Intelligence
www.elsevier.com/locate/artint

AI and Law: A fruitful synergy

Edwina L. Rissland [a,*], Kevin D. Ashley [b], R.P. Loui [c]

[a] *Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA 01003, USA*
[b] *Graduate Program in Intelligent Systems and School of Law, University of Pittsburgh, Pittsburgh, PA 15260, USA*
[c] *Department of Computer Science, Washington University, St. Louis, MO 63130, USA*

Received 29 July 2003

AI and Law is a classic field for AI research: it poses difficult and interesting problems for AI, and its projects inform both AI and its focal domain, the law itself. This special issue reports on a range of projects, from those where the law motivates fundamental research and whose results reach beyond the legal domain, to those that partake of the benefits of techniques and wisdom from AI as a whole. For instance, projects tackling legal argument have not only created programs that produce legal arguments but also led to insights and advances in the logic of argumentation. Projects with an applications bent have often provided insights about the limitations and nuances of existing techniques, and have served as the catalysts for developing new approaches. For instance, harnessing models of legal argument to teach law students how to argue has led to refinements to and extensions of the models. There is a synergy not only between law and AI, but also between AI and AI and Law. In fact, the work on Case-Based Reasoning (CBR) done in the AI and Law community provided one of the most important streams of results that contributed to the birth of that area in the mid-1980s. Currently, work on legal argumentation is having a similar impact on the international non-standard logic and argumentation communities.

AI and Law is much more than an applications area. Its concerns touch upon issues at the very heart of AI: reasoning, representation, and learning. For the AI researcher interested in symbolic methods—or methods of whatever stripe—that are focused on providing explanations and justifications, AI and Law is an excellent arena. No matter how a reasoner arrives at a legal answer it must be explained, justified, compared to and contrasted with alternatives. For the researcher interested in topics like negotiation, decision-making, e-commerce, natural language, information retrieval and extraction, and data mining, AI and Law is a rich source of problems and inspiration.

* Corresponding author.
E-mail addresses: rissland@cs.umass.edu (E.L. Rissland), ashley@pitt.edu (K.D. Ashley), loui@cs.wustl.edu (R.P. Loui).

0004-3702/$ – see front matter
doi:10.1016/S0004-3702(03)00122-X

AI and Law has implications for fields beyond computer science and law. Fields like ethics, with law-like structures—norms, sources of authority, reservoirs of prior experiences—and with important societal overtones, are obvious candidates. Fields like psychology and philosophy, with a deep concern for the nature of concepts, prototypes and analogy, share many core concerns with AI and Law. In the best AI tradition, AI and Law seeks to provide concrete computational means to challenge assumptions and provide grounding on many questions of jurisprudence, and while it is respectful of the wisdom of legal scholars, it by no means accepts it on faith. AI and Law remains a place for scholars; no matter what one's philosophy of law, science, or mind, AI and Law provides a vital forum for debate and analysis.

This issue is the culmination of an effort that began at the biennial International Conference on AI and Law (ICAIL-01) held at Washington University in St. Louis in May 2001. It presents fully mature projects as well as those in active development. Papers in this special issue, some of which grew out of conference papers presented at ICAIL-01, were reviewed by experts in AI and Law as well as those in the field of AI at large. The editors encouraged presentations that were readily accessible to those outside the specialty, for instance, by requesting authors to include illustrative examples. One need not feel that a legal background is necessary to understand the papers.

The rest of this editors' introduction comprises three sections: a short discussion about the law and why it is an interesting and challenging domain for AI, a very abbreviated account of the 30-plus year history of AI and Law, and a brief introduction to the papers.

1. The nature of law

The legal domain has many characteristics that make it especially interesting and challenging for AI:[1]

- *Diverse categories of knowledge.* Law has an abundance of cases, rules, theories, procedures, hierarchies of authority, norms and meta-rules. Cases include actual precedents, that is, fact situations that have been litigated and decided at the trial court level, and whose decisions may have been appealed up through various appellate court levels of the judicial system. *Hypothetical cases* ("hypos") often arise in courtroom oral arguments and Socratic classroom interchanges. *Prototypes*, often streamlined

[1] While the Editors acknowledge their bias of thinking of the law in terms of American law, most of these remarks apply across the vast spectrum of legal systems. Many good books and articles discuss legal reasoning, its nuances, and problems: *An Introduction to Legal Reasoning* by Edward H. Levi (University of Chicago Press, 1949), *The Bramble Bush* by Karl N. Llewellyn (Oceana, 1930), *How To Do Things With Rules* by Twining and Miers (Weidenfeld & Nicolson, 1976), *The Nature and Functions of Law* by Berman and Greiner (Foundation Press, 1980), *The Civil Law Tradition: An Introduction to the Legal Systems of Western Europe and Latin America* by J. H. Merryman (Stanford University Press, 1969), *The Case Law System in America* by Karl N. Llewellyn (translated by Ansaldi, edited by Gewirtz, University of Chicago Press, 1989), "Romantic Common Law, Enlightened Civil Law: Legal Uniformity and the Homogenization of the European Union" by Vivian Grosswald Curran (*Columbia Journal of European Law*, Vol. 7, 2001). Several of these have gone through many editions and are currently available in paperback.

hypos, capture the essence of a problem or idea. Rule-like knowledge includes statutes and codes, constitutional principles, norms of interpretation, rules of criminal and civil procedure, heuristic rules of thumb, "black letter" generalizations abstracted from many actual precedents, "rules of a case" summarizing the main conclusion, or *holding*, of a case. Legal systems include hierarchies of common, statutory and constitutional law (e.g., the Constitution prevails over statutes, statutes prevail over court-made rules, state statutes prevail over local ordinances) and a hierarchical system of courts with varying degrees of authority (e.g., appellate courts can review and prevail over trial courts). In the United States state and federal law fall under (nearly) separate hierarchies. In Europe, besides the laws of individual nations, there are those of the EU.

- *Explicit styles and standards of justification.* In Anglo-American law *stare decisis*—the doctrine of precedent—governs much legal reasoning. *Stare decisis* requires that similar cases be decided similarly. While this doctrine puts the focus squarely on reasoning from case to case, it is silent on how "similarity" should be determined. In fact, similarity is not static; it can depend on one's viewpoint and desired outcome. In civil code countries and jurisdictions, like France, Germany and Japan, the style of reasoning places more emphasis on reasoning with rules and codes. Of course, courts in common law legal systems also reason about statutory rules and codes; however, they give a more important role to cases in statutory interpretation than do courts in civil code countries. Whatever the particular standards of justification and explanation, the fact that the standards are explicit helps make legal systems more socially accountable.
- *Different modalities of reasoning.* In concert with different types of knowledge, there are different types of reasoning, for instance, reasoning with cases alone, rules alone, cases and rules together, etc. As one delves further into a single modality, one often discerns that the reasoning is hybrid in nature. For instance, in reasoning with rules, one must often resort to reasoning with cases to handle gaps, conflicts, and ambiguities in the rules and their constituent terms.
- *Specialized repositories of knowledge.* Large collections of cases are available from a great variety of courts. In the United States, the courts of general jurisdiction include state and federal trial courts and appellate courts, including state supreme courts and the United States Supreme Court. Several specialized courts handle areas like admiralty, bankruptcy, and tax law. Even hundreds of years after the fact, cases are available for use in analysis and, if still good law, argument. Also important are constitutions, federal and state statutes, local ordinances, and other charters, rules, and regulations of governance. Legal knowledge in the form of cases and statutes is highly "vascularized" in the sense that items are extensively cross-linked.[2] Commercial

[2] Cases point backwards to authorities, typically precedent cases, through citations, which come in many types and are often introduced by so-called *citation signals*, such as *See* that indicates an authority clearly supports a legal proposition, *But see* that indicates support of a contrary proposition, *Cf.*, that indicates support of a different but sufficiently analogous proposition, etc. These are set forth in style books such as *The Bluebook* published by the editors of the Columbia, Harvard, University of Pennsylvania, and Yale law journals. Citation services like Shepard's compile "forward pointers" to subsequent cases that cite the given case.

services like Shepard's Citations keep track of such linkages—both forward and backward—according to a rich taxonomy of link types. There are many secondary sources like scholarly treatises, restatements, commercial summaries, casebooks, and practice guides. Sources like the restatements are compendia of generalized rules (so-called "black letter" rules) and illustrative real and hypothetical cases that committees of legal scholars have compiled through examination of actual cases. While most secondary sources do not have the authoritative status of primary sources, like enacted statutes and actual precedents, some are accorded considerable respect. Some sources, like commercial summaries in important areas like tax law, are constantly updated; others, like scholarly treatises and the *Restatements*, are revised on a much longer time scale. All of these give the law a vast, long-lived, and dynamic "institutional memory"; much of this, including the linkages, is available on-line.

- *A variety of task orientations:* There are a variety of task orientations such as *advocacy, adjudication, advising, planning and drafting,* and *administration.* Underlying all of these tasks is legal *analysis* of the facts and circumstances and how they relate to relevant law. In *advocacy*, one takes a "side" in a controversy and argues for an outcome or interpretation that favors it. In *adjudication*, a court (i.e., a single judge or a panel) decides a controversy and usually publishes an opinion that puts forth the reasons for the decision. In *advising*, a lawyer examines a legal situation, typically while it is still evolving and before it becomes a full-blown dispute, and offers advice about alternative courses of action and their (legal) benefits and risks. *Planning* includes structuring contracts, negotiating commercial "deals", developing estate plans, and setting up charitable trusts. *Drafting* includes creating the documents needed to implement them, as well as the formulation and writing of statutes and other forms of legislation. *Administration* includes the application of governmental regulations and policies, such as those governing income taxes or social security benefits, by officials and governmental agencies. Currently, many AI and Law efforts focus on advocacy. Very few, if any, tackle adjudication. AI and Law researchers may be bold but they are seldom vainglorious; they know that judging is an exceedingly complex and nuanced task involving the law's role in society at large.[3] A long line of scholars—including Oliver Wendell Holmes, Karl Llewellyn and Jerome Frank—known as the American Realists, has addressed the many philosophical, sociological, psychological, and economic aspects of judging; currently these issues are the focus of scholars from disciplines like Law and Economics and Critical Legal Studies. Advice-giving, planning, and document drafting are the day-to-day focus of most practicing lawyers and the topics addressed touch every aspect of life, including death and taxes. Interestingly, these tasks, for the most part, have received little attention from AI and Law researchers. Administrative law has been explored for the most part by those interested in rule-based (e.g., logic programming) approaches, and several systems have been fielded, particularly in Europe and Australia.

[3] Contrary to some popular notions, law is *not* a matter of simply applying rules to facts via *modus ponens*, for instance, to arrive at a conclusion. *Mechanical jurisprudence*, as this model has been called, is somewhat of a strawman. It was soundly rejected by rule skeptics like the realists. As Gardner puts it, law is more "rule-guided" than "rule-governed".

- *Open-textured concepts.* Concepts in the law are not black and white with hard-edged boundaries between positive and negative instances. While there are central prototypical cases whose classifications seem clear, concept boundaries are gray zones and contain cases with viable competing interpretations. Legal concepts are more like "open" sets than "closed" sets, to use mathematical terms. The legal philosopher H.L.A. Hart characterized legal concepts as having a "core of settled meaning" and a "penumbra". Legal concepts, therefore, cannot be modeled by unassailable, universally quantified necessary and sufficient conditions. In a word: they are incurably *open-textured*. Furthermore, legal concepts and the law as a whole evolve. In response to new situations that require resolution, concept boundaries are refined and pushed around, and exceptions are carved out, even from the core itself. While this makes legal concepts seem like blurry-edged Swiss cheese, it also enables the law to respond to societal change. By contrast, if everything were static and totally clear, one could hazard to say that the law would be dead. Unlike mathematics, as a result of open texture and other reasons, legal questions (e.g., what speech is protected by the First Amendment?) have competing, reasonable "answers" and they too evolve.
- *Adversarial truth seeking.* The underlying expectation in the legal domain is that, through the adversarial process, ideally the truth will out. For the law, adversarial argument is a feature, not a bug: it is the anvil on which truth is hammered out. Legal argument can be viewed as an exercise in competitive theory formation: each side forms its theory using cases and other information that support its desired conclusions while at the same time minimizing, distinguishing, undercutting or avoiding the pitfalls of the opposing theory. Through vigorous debate between parties espousing competing interpretations and outcomes, the pros and cons emerge. Although one will ultimately prevail, it does not necessarily mean that the alternatives were without value. If they had been, there would not have been a debate about the issues (e.g., the parties would have settled). Rather it means that the interpretation that prevailed was more persuasive to the authority—the court—that decided the issue. This means that legal reasoning is "two handed"—on the one hand, on the other hand—and that defeasible reasoning and dialectical argument are important in law.
- *Highly reflective.* The law is a very reflective intellectual discipline. It constantly examines and re-examines its underlying methods and missions. Jurisprudence, the philosophy of law, is an active area of scholarship. In providing computational models that address how one thinks in legal matters, AI and Law seeks to provide an alternative grounding for the analyses of jurisprudential scholars and new tools for investigating their ideas. Models of argument and case similarity are good examples of how AI and Law research can flesh out skeletal descriptions developed by legal scholars, and can provide a way to explore them to see how they work and how well they work. AI and Law has been described by the late Donald Berman, one of the field's founders and a distinguished professor of law, as "a new analytical jurisprudence".

While these characteristics suggest the legal domain's richness, they also reveal the promise of synergy between law and AI. The law is a domain intermediate between the mathematical/scientific domains AI has tackled, and the domains of everyday experience it hopes to tackle. The legal domain deals with real world scenarios involving all aspects of

meaning of words. It drew attention to the fact well-known in the law that one cannot reason by rules alone, and that in response to failure, indeterminacy, or simply the desire for a sanity check, one examines examples. In Gardner's system, which analyzed so-called "issue spotter" questions from law school and bar exams in the offer-and-acceptance area of contract law, the examples were not actual specific precedents but general, prototypical, fact patterns. Her work sought a principled computational model of the distinction between "hard" and "easy" cases, much discussed in jurisprudence.[8] She framed her discussion in terms of defeasible reasoning, a topic of intense interest today.

While progress continued on rule-based reasoning (RBR) systems in the 1980s, there began to emerge a community of AI researchers who focused on reasoning with cases and analogies—that is, case-based reasoning. In the early 1980s, Rissland had investigated reasoning with hypothetical cases particularly in Socratic law school interchanges. In 1984, she and Ashley first reported on the legal argument program HYPO and the mechanism of "dimensions". This line of research had grown out of Rissland's earlier work on example-based reasoning and "constrained example generation" in mathematics.[9] Initially concerned with the problem of generating hypotheticals (hence its name), HYPO reached full maturity as a case-based argumentation program in Ashley's doctoral dissertation. It was the first true CBR system in AI and Law, and one of the pioneering systems in CBR in general. Thus by the mid 1980s, RBR and CBR approaches were making themselves felt in AI and Law.

In her excellent review article, Anne Gardner points out that this bifurcation between rule-based and case-based approaches is longstanding. We note that often champions of one approach appreciate full well the importance or need for the other (e.g., Buchanan), switch their focus (e.g., McCarty), seek to bridge the gap between them (e.g., Gardner), attempt to reconcile them through reconstruction (e.g., Prakken, Sartor and Bench-Capon), or are intrigued by hybrid approaches (e.g., Rissland).

In the mid 1980s, a few leading American law schools began conducting seminars on AI and Law. The first was given at Stanford Law School in 1984 by three law professors: Paul Brest (later to become Dean), Tom Heller and Bob Mnookin. Rissland launched her seminar on AI and Legal Reasoning at the Harvard Law School in 1985, and Berman and Hafner theirs at Northeastern in 1987. Over the years, such seminars have proliferated and have served as forums bringing together the AI and legal communities.

---

[8] If all the experts consulted on a question agree as to its interpretation, it is *easy*; otherwise, it is *hard*. Cases that get settled before they are litigated are typically easy, and those that become court cases, especially those that make their way up the appellate ladder, are hard. Those that end up in the Supreme Court are *very* hard. One should note common wisdom says that 80% of all disputes are settled "on the court house steps", that is, before they go to trial.

[9] It is interesting to note that Rissland first reported on her "constrained example generation" (CEG) model that used a "retrieval-plus-modifications" approach to generate counter-examples in mathematics at the same 1980 conference, The Third Biennial Conference of the Canadian Society for Computational Studies of Intelligence, at which McCarty and Sridharan reported on their "prototypes-plus-deformations" model of legal argument. Theirs were back-to-back papers in the same session. CEG was in fact an early example of adaptive CBR: retrieve a good (enough) example that matches as many of the desiderata as possible from an Examples-space (a Case Base) and then try to satisfy other goals with modifications.

The 1980s saw a significant ramping up of interest in AI and solidifying of the research community. A few specialized conferences, such as those at the IDG in Florence and at the University of Houston were followed by an IJCAI-85 panel of AI and Law researchers aimed at a general AI audience.[10] The founding of the Computer/Law Institute at the Vrije Universiteit in 1985 by Guy Vandenburghe, at which AI and Law research was subsequently directed by Anja Oskamp, led to research groups throughout the Netherlands. Research on AI and Law in Japan began at this time as well in Hajime Yoshino's lab at Meiji Gakuin University in Tokyo. Japan's Fifth Generation Computer System Project (1982–1995) provided great impetus, particularly in the use of expert systems and other logic-based techniques.

All this happened before 1987, a watershed year in AI and Law. The first International Conference on AI and Law (ICAIL) in 1987, organized by Carole Hafner and Don Berman, was held at Northeastern University, where they had just established a center for Computer Science and Law. Since then these biennial conferences have served as the anchor and showcase for the entire community.[11] This marks the beginning of what we might call the contemporary era of AI and Law. After the second ICAIL meeting in 1989, a committee was formed to develop a charter for an international organization. This led to the founding of the International Association for AI and Law in 1991. The journal *Artificial Intelligence and Law*, the journal of record for the AI and Law community, made its debut in 1992.[12] Its special issues provide excellent snapshots of progress in an area. A recent triple issue in memory of Donald Berman, one of the field's leading lights and most beloved members, contains articles by many of the field's stalwarts (McCarty, Ashley, Rissland, Sartor, Bench-Capon, Prakken), as well as a paper by Hafner that coalesces and updates her three ICAIL conference papers with Berman that remain among the crown jewels of the field. In the same issue, McCarty reports on his program to use deontic logic (the logic of permissions and obligations, rights and duties most closely associated with the famous Yale legal scholar Wesley Hohfeld[13]) to represent difficult legal concepts like ownership and shed light on this topic of longstanding interest in jurisprudence.

It was also in 1987 that MIT Press published Anne Gardner's *An Artificial Intelligence Approach to Legal Reasoning*, a revision of her 1984 Stanford Ph.D. dissertation.[14] It was

[10] The members of the panel were Edwina Rissland (Chair), Don Waterman, Anne Gardner, Thorne McCarty, Kevin Ashley, and Michael Dyer. They discussed many of the enduring issues—like open texture and the complementarity of CBR and RBR—still of interest today. They were cautious about the creation of intelligent aids for legal practitioners.

[11] ICAIL-87 was in Boston; ICAIL-89, Vancouver; ICAIL-91, Oxford; ICAIL-93, Amsterdam; ICAIL-95, College Park, Maryland; ICAIL-97, Melbourne, ICAIL-99, Oslo; ICAIL-01, St. Louis; and ICAIL-03 is scheduled for Edinburgh. Proceedings from the conferences are available from the ACM.

[12] Volume 1, No. 1 contained articles on a theory of case-based argument (Skalak and Rissland), deontic logic as a representation of law (Jones and Sergot), legal knowledge-based systems (Bench-Capon and Coenen), legal practice systems (Lauritsen), and a review of Ashley's book. Such an interesting mix of topics is typical.

[13] His ideas appeared in a pair of *Yale Law Journal* articles in 1913 and 1917.

[14] The initial chapters provide an excellent introduction to the jurisprudence of open texture, rules, legal positivism, judicial discretion, etc., and the positions of the legal theorists (e.g., Hart, Llewellyn, Fuller, Dworkin) most closely associated with these topics. Ron Loui's "Hart's Critics on Defeasible Concepts and Ascriptivism" in the proceedings of ICAIL-95 gives a detailed and scholarly discussion of Hart and defeasibility.

the first of two very influential Ph.D. theses published in the short-lived MIT Press series on AI and Law, edited by McCarty and Rissland. The second was Kevin Ashley's *Modeling Legal Argument* (1990), his dissertation done with Rissland in her lab at UMass/Amherst. In 1987, Oxford University Press published Richard Susskind's book *Expert Systems in Law*, based on his doctoral work; it had a wide influence in Europe.

Ashley's dissertation, completed in 1988, presents a model of legal argument in which reasoning with concrete cases—that is, *actual* appellate precedents—is paramount. HYPO produced point-counterpoint style arguments in the area of trade secrets law. It provided a detailed model of many of the key ingredients of the Anglo-American doctrine of precedent (*stare decisis*): how to assess relevancy, compare cases, analogize and distinguish cases using relevant similarities and differences. HYPO has had many progeny. One of the many systems, Vincent Aleven's CATO system, described in this special issue, teaches law students *how* to create case-based arguments. At its core are "factors", a mechanism deriving from HYPO-style "dimensions". The wide influence of the HYPO/CATO theory is manifest in the range of work that uses arguments generated by these systems as benchmarks for evaluation of other systems and theories. The paper by Bench-Capon and Sartor in this issue is a case in point.

Also in 1988, the first Jurix conference was held in Amsterdam, reflecting the growing activity in the Netherlands, particularly in knowledge-based systems. The first conference was a purely local event, but these annual conferences rapidly acquired an international flavor, and since 1990 have provided an important forum for European researchers. The European dimension was completed in 2002 when Jurix was held in London, the first time it had left the Netherlands and Belgium. In 1988, the American Association for Artificial Intelligence (AAAI) founded a subgroup in Law at the urging of Rissland who then served as Liaison; the subgroup was in existence for about ten years until its function was largely replaced by the International Association for AI and Law.

In 1990, the *Yale Law Journal* published Rissland's article "Artificial Intelligence and Law: Stepping Stones to a Model of Legal Reasoning". It both provided a progress report and made the case to the legal community of the interest and importance of work done in the area. In June and July 1991, a pair of special issues of the *International Journal of Man-Machine Studies*, edited by Rissland, showcased many of the boldest projects of the day: HYPO, CABARET, GREBE, SCALIR, and PROLEXS, as well as two early papers by Tom Gordon and by Trevor Bench-Capon and Tom Routen on formalizing legal argument using techniques from logic. These latter two papers marked the beginning of a stream of research on argument that has grown into a torrent in recent years.

A few papers in these *IJMMS* issues explored reasoning with cases in concert with reasoning with rules.[15] Rissland and her student David Skalak reported on CABARET, the first truly hybrid CBR-RBR reasoner; it used an agenda-based architecture to integrate classic rule-based reasoning and HYPO-style CBR in the statutory area of US tax law

[15] One can view Gardner's program in this way also: for instance, the program reasons with 'cases' when the rules run out. However, her cases were not concrete cases (i.e., actual legal precedents) as they are in CABARET and to a large extent in GREBE, which also includes prototypical cases that are not precedents. An earlier doctoral project at MIT by Jeffrey Meldman in 1975 on the law of assault and battery also used rules and cases, but the cases were represented as rules that encoded their *rationes decidendi*.

concerning the home office deduction. From the viewpoint of AI, the project sought to investigate the architecture and control issues needed to use CBR and RBR in concert to complement and supplement each other; CABARET did not simply call one serially after the other, but instead dynamically and opportunistically interleaved them. From the viewpoint of law, the project sought to explore ways to operationalize a theory of statutory interpretation that intertwines reasoning with cases and reasoning with rules in a three-tiered theory of argument strategies, moves, and primitives. The agenda was controlled by heuristic rules that embodied the theory: for instance, if all but one of a rule's prerequisites are satisfied (i.e., there is a rule-based "near miss"), CABARET used cases to argue that the predicate had actually been satisfied, or alternatively, that it was not necessary. Rissland and Skalak purposely chose to address a reduced version of the problem of statutory interpretation that explicitly did not consider policies, principles and other important normative considerations.

Karl Branting's paper in the same issue described how structure mapping from cognitive science and classic A* search from AI are used in GREBE, his program that could re-use existing arguments and portions of them[16] to generate arguments for new cases in the domain of Texas worker's compensation law. GREBE can also be viewed as a hybrid CBR/RBR program since it reasons with both rules and cases. For instance, it creates (structural) analogies when the rules run out or are otherwise inconclusive to show a legal predicate has been satisfied. In our special issue, Branting uses his experience with GREBE to elucidate key aspects of legal argument structure having to do with "warrants", an idea originating in Toulmin's classic work. There was also some exploration of hybrid systems using blackboards (e.g., PROLEXS) and sub-symbolic connectionist models (SCALIR). These veins have not been much emphasized, although interest in such approaches regularly resurfaces.

The theme of heuristic search in the service of argument creation became the focus of another project by Rissland and Skalak. Since the space of information is far too vast to explore profligately, and since one cannot conceivably use everything discovered, there is a need to control the search for and the accumulation of pieces of information. The BankXX project explored how knowledge about an evolving argument—its growing collection of supporting, contrary, leading, best cases, legal theories, prototypes, etc.—could guide a program to uncover and harvest information using best-first heuristic search of a space of legal knowledge (about personal bankruptcy law) that included legal cases, citations, domain factors, theories, and stories.

Since its debut, hybrid CBR-RBR has been explored by others, most notably John Zeleznikow, Andrew Stranieri, George Vossos, Dan Hunter and their colleagues in Australia, who have built several hybrid systems. Ikbals, built by Vossos in conjunction with his 1995 Ph.D. at La Trobe University in Melbourne, was a CBR-RBR hybrid with machine learning capabilities that operated in the law of loans provided by financial institutions. Split-Up integrated neural nets and rules to determine property divisions in divorce settlements. Also, in the mid 90s, the HELIC-II project of Katsumi Nitta and

[16] These are called *exemplar-based explanations* or *EBEs*.

Masato Shibasaki, Tokyo Institute of Technology, combined RBR and CBR, and explored defeasible and dialectical reasoning.

By the mid 1990s, the field was clearly well on its way to tackling some of the central issues in legal reasoning: reasoning with rules (especially in the face of conflict between rules), reasoning with cases, and open texture in legal predicates. There was now a well-established international community; several members of the second generation had fledged and left their Ph.D. institutions to establish their own research programs; and ideas were being explored by others than their originators.

The 1990s saw a renewal of interest in legal information retrieval, in part because of improved retrieval engines, new learning-based information extraction techniques, and the dramatic rise of the World Wide Web. For instance, in 1995, Hafner edited a special issue of *Artificial Intelligence and Law* devoted to intelligent legal text-based systems.[17] Conceptual retrieval and the automation of the representation of legal sources have long been goals of AI and Law research. This focus was represented in that issue by the Flexlaw system of J.C. Smith and his group at the University of British Columbia. Also in that issue Graham Greenleaf and his colleagues made some pioneering reflections on the relationships between knowledge-based systems, databases, and hypertext systems. With the advent of the WWW this became a very important topic. Greenleaf and his group founded the Australasian Legal Information Institute (AusLII) in the mid 1990s to further develop this work. Subsequently there have been additional efforts in this direction both in Europe and the US. For instance, Marie-Francine Moens explored the use of AI techniques for automatic text processing for IR in her 1999 doctoral dissertation at the Katholieke Universiteit Leuven, Belgium. In the mid 1990s, Rissland and her doctoral student Jody Daniels developed SPIRE, a system that used results of HYPO-style CBR analysis to drive a full text retrieval engine that operated at two levels: to retrieve cases (i.e., full text opinions), and within individual cases to retrieve passages. In the late 1990s, Ashley and his student Stefanie Brüninghaus developed SMILE, a system that employed learning-based techniques to extract information about factors from full text sources. The article by Jackson et al. in this issue continues in this vein; it shows how progress is being made in an applications context.

In Europe, conceptual retrieval, principled systems development, and sharing and reuse of knowledge based on ontologies were given additional impulse by the need to harmonize legislation across the polyglot countries of the European Union. Initiated through the Ph.D. work of Andre Valente at Amsterdam and Robert van Kralingen and Pepijn Visser at Leiden, and further developed by Joost Breuker and Radboud Winkels at the University of Amsterdam, and by Visser and Bench-Capon at the University of Liverpool, legal ontologies are the focus of much activity and the subject of regular workshops.

Since the 1990s, a burgeoning community has focused on developing models of argumentation. Some researchers like Giovanni Sartor and Ron Loui have concentrated on models that address reasoning with norms. In the mid 1990s, Tom Gordon developed a dialogue-based model of legal pleading, and Loui and Norman developed a discourse-

[17] The lead article by Howard Turtle, who has done much foundational work in IR, provides a good introduction to the issues.

based model of legal argumentation.[18] Loui and Norman focused on defining categories of rationales used in adversarial arguments, such as "compression" of complex, specific rules into simpler, general ones, and the forms of attack appropriate for each. Gordon developed his dialogue approach into the web-based ZENO system, which was used in Germany for facilitating public commentary on a planned high technology park and residential zone, and Loui developed his web-based "Room 5", which allowed users to argue about US Supreme Court cases involving such issues as freedom of speech. Others who have sought to provide logic-based models of legal argument include Jeff Horty and many in the Japanese AI and Law community, including Katsumi Nitta and Hajime Yoshino. Several doctoral theses and subsequent books from the energetic Dutch community, such as those of Jaap Hage, Henry Prakken and Bart Verheij, have made significant strides on developing models of argumentation.[19] Many of these projects have dealt with the sort of arguments performed by HYPO and its progeny. Over the years, work in this area has only intensified; hardly an issue of *Artificial Intelligence and Law* is published without some article addressing this topic. In 1996 a special double issue of *Artificial Intelligence and Law,* edited by Prakken and Sartor, was devoted to logical models of legal argumentation, and in 2000, another special double issue, edited by Feteris and Prakken, focused on formal and informal models of dialectical legal argument. In our special issue, the article by Bench-Capon and Sartor presents a well-developed theory of case-based legal argument that involves the use of HYPO-style dimensions/CATO-style factors as well as norms. The article by Verheij shows how ideas about dialectical argument can be used to build environments to assist in the creation of arguments.

3. Introducing this issue

This special issue of *Artificial Intelligence* presents a sampling of current work in the field of AI and Law. Many of the papers continue to work veins of research mined since its birth, and also of long-standing centrality to the law itself. For instance, in the first article, Ashley and Rissland address issues at the core of legal reasoning, particularly how the law reasons with cases and analogies, how as a system the law can be said to learn, and the relationship between representing relevant similarities among cases and modeling learning.

Branting's article integrates his previous work on GREBE with his work on representing the justifications of legal decisions—the so-called *rationes decidendi.* He focuses on showing how the structure of a legal decision affects its use in analyzing and deciding subsequent cases. Branting proposes modeling decisions as reduction graphs in which

[18] For example, see Loui and Norman's "Rationales and Argument Moves" in *Artificial Intelligence and Law* (Vol. 3, No. 3, 1995) and Gordon's *The Pleadings Game—An Artificial Model of Procedural Justice* (Kluwer, 1995).

[19] Hage, *Reasoning with Rules, An Essay on Legal Reasoning and its Underlying Logic* (Kluwer, 1997). Prakken, *Logical Tools for Modelling Legal Argument—A Study of Defeasible Reasoning in Law* (Kluwer, 1997). Verheij, *Rules, Reasons, and Arguments: Formal Studies of Argumentation and Defeat* (Dissertation, U. Maastricht, 1996).

each reduction operator corresponds to a justification step, or warrant, available for use in arguments about other cases. This model shows how a decision's justification, not just its facts and outcome, influences how it can be used to make arguments about other cases. It also permits portions of multiple decisions to be combined to form new arguments.

Bench-Capon's and Sartor's article develops a logically-grounded account of how cases are used in legal reasoning, particularly for defeasible reasoning. According to their view, reasoning with cases is a process of theory construction, evaluation, and application. They provide a definition of what constitutes a theory of a body of case law, and how competitive theories are constructed. Their work is a capstone of a line of inquiry by the logic-oriented sub-community of AI and Law that has long pursued an agenda to describe in logical terms HYPO/CATO-style reasoning: in particular, the role of factors and dimensions, and to incorporate it in a regime that includes rules and norms. Their approach also gives a central role to a notion of the purposes motivating legal theories, revealed in cases and used to ground preferences between rules.

McLaren shows how the idea of A* search used so potently in GREBE can be coupled with new insights on how to "operationalize" norms and past rationales for analyzing new fact situations. He does this, not in the law, but in ethics, a related normative domain full of open texture, precedents, and hard questions but offering somewhat less structure and constraint than law for AI techniques to employ. In his SIROCCO system, McLaren's refined approach to structure mapping and A* search includes a more nuanced assessment of matching that takes into account multiple levels of representation. He evaluates these ideas in the context of retrieval.

Aleven presents the definitive report on his CATO project. CATO harnesses key mechanisms of HYPO-style reasoning to teach law students how to make good precedent-based arguments. He enriches the underlying HYPO model by refining and extending its representation and use of factors, by focusing on representing the reasons why factors matter as relevant similarities or differences among cases. He evaluates both the contribution these reasons make to argument quality and how well CATO teaches law students to make such arguments. In the movie *The Paper Chase*, the character of Professor Kingsfield, a stereotypical curmudgeon of a law professor, throws down the gauntlet to his fresh-faced 1L's (first-year students) by announcing that, "You come in here with a head full of mush and you leave thinking like a lawyer." Aleven's CATO shows us a way—a gentler if not better way—to accomplish this.

Jackson and his colleagues give us a window on the use of intelligent information retrieval and extraction in the legal domain. They report on their efforts to apply information extraction techniques to full text court opinions in order to ferret out the linkages between cases. Linkages are used to identify and summarize the (procedural) history of an individual case as it makes its way through the court system, and to discern how a case is commented upon and viewed in subsequent cases that discuss it and how it interprets prior cases that it cites. As anyone with even the most casual contact with natural language understanding knows, this is far from a simple task, since cases can be cited and referred to in a daunting variety of ways, and from widely disparate portions of a text. Developing a system that once trained does this automatically and to a level that meets commercial standards is a challenge.

The paper by Verheij is one of several recent attempts to implement and explore the consequences of new theories of argument. Verheij explores the use of two theories of argument: one that emphasizes the role of support and undercutting, and a second that allows for argumentation about warrants and the "defeaters" themselves. He demonstrates how ideas about informal logic—defeasible reasoning in particular—can contribute to a technology for those who need to author, analyze, visualize, or collaborate in making disputational arguments. As history shows, pictorial devices and logic games have been important in logic, especially at times of intellectual revolution.[20] Work like Verheij's strives to show how a new understanding of dialectic can improve on the standard Toulmin diagram. While lawyers and law students may not need such tools, untrained persons who need to understand the law may benefit. The successful schematic has explanatory value and imposes discipline. Indeed, something like Verheij's system may one day be available in popular software, just as grammar-checking has become a standard in word processors.

In conclusion, in an age when lay people have ever-growing access to legal data and when their interactions with an e-bureaucracy are more frequent and more subject to automated regulation, the real impact of AI and Law may be on ordinary citizens and the kind of computerized society they live in. What kinds of e-commerce contracts will be negotiated by agents, and what sort of grievance procedures will be available when things don't work out as hoped? What kinds of electronic billing, securities oversight, tax collection, profiling, etc. will there be? These questions are difficult even when they involve human experts; what are the prospects when they involve artificial agents? Lawyers and ordinary citizens alike are rightly suspicious of the dictates of naively interpreted rules, and our legal systems have worked long and hard to incorporate nuanced reasoning that takes into account the special circumstances of individual cases as well as the larger goals and purposes of society. In developing sturdy and stable social systems suitable for automation, people may want their electronic societies, or the electronic faces of their institutions, to offer the protections of the legal system's complex structure and interpretive traditions. That kind of knowledge can only come from progress in AI and Law.

Acknowledgements

The editors appreciate the assistance they have received from Carole Hafner, Anja Oskamp, and others in the AI and Law community in preparing this article. In particular, they thank Trevor Bench-Capon for his ever thoughtful comments. They are especially grateful to Anne Gardner for her extensive and detailed comments. They, of course, acknowledge the abundant effort of all those who submitted articles for this special issue and those who reviewed them.

[20] Martin Gardner wrote the book *Logic Machines and Diagrams*, Layman Allen developed the game *WFF'n Proof* to teach logical reasoning, and proper intellectual historians cite Peirce's circles alongside Frege's strokes.