Intellectual Artefacts of Expert Systems Meta-epistemology*

Pamela N. Gray
PhD Candidate, School of Law
University of Western Sydney
Parramatta, NSW, Australia,
Tel 61 02 9872 255, www.grayske.com, email: pgray@csu.edu.au

Abstract

Essential to intelligent programs is computational epistemology. Expert systems, derive their computational epistemology from domain expertise. Construction of an expert system shell and an application requires a meta-epistemology that transforms domain epistemology through a sequence of computational epistemology, shell epistemology; programming epistemology and application epistemology (as distinct from application ontology) into an expert system. This paper explains the meta-epistemological method by reference to prototypical development of the eGanges shell (2002-4) that is suited to domains (such as law) with rule systems, systemic procedures and/or strategic paths (RPS). The computational epistemology of 3d logic is used as its logic reification suits object-oriented programming. Retroduction (Peirce, 1931. p.28), commonly known as abduction, used according to common expert sense, effects the transformation. The metamorphosis is achieved by a sequence of selections of what fits next, and what needs to be repositioned for better fit. In the eGanges design, a central two dimensional tributary structure called a river or rule map is selected out of the 3d logic model, to optimize cognitive value of the computational epistemology for the user interface. River ideographs are streamlined flowcharts that resemble fishbone diagrams. Extensive, dense river ideographs, may be variously nested as sub-maps, and also variously glossed to incorporate annotations of the RPS system. Gloss options include links between nodes in the same set of sub-maps and between parallel river systems. Sub-epistemologies may be required for the glosses. Functionality of the shell facilitates navigation and interrogation of the maps, as well as processing interrogation input.

Keywords: 3d logic, abduction, adversarial logic, alternative rule sets, combinatorial explosion, combinatorial implosion, common expert sense, communication system, domain epistemology, eGanges, epistemology, expert-friendly, expert system, Feigenbaum bottleneck, fishbone, formalisation, functionality, glosses, graphical logic, heuristics, interface, knowledge engineering, knowledge representation, large scale applications, meta-epistemology, navigation, nesting, object-oriented, parallel maps, pole meta rules, procedure maps, logic reification, retroduction, river logic, rule maps, spectra, standardisation, strata logic, strategy maps, sub-maps, temporary negative, temporary uncertainty, transparency, tributary structures.

1 Meta-epistemology

At the outset of development of expert systems technology, McCarthy and Hayes (1969) laid down one of the requirements for validity of a system: the epistemology had to be adequate. A definition of this adequacy was given (Lifschitz, 1990, p.3) as follows:

A representation is epistemologically adequate if it can be used to express the facts that can actually be discovered with the available opportunities.

Samuel (2003) recognised the need to find legal epistemology for the development of artificial legal intelligence. The author arrived at this conclusion independently in 2003, by recognising that it was a computational domain epistemology of artificial legal intelligence, the model of 3d legal logic (Gray, 1990, 1995, 1997) that she had developed a decade earlier.

For the purposes of expert system development, the expert epistemology is the way that the expert derives new information from available information. The delineation and operation of the idea systems of experts that enable them to derive new information is a matter of domain epistemology that may be construed in a computational form. Further adaptation of computational epistemology for the purposes of program requirements may be necessary to suit user specifications for the expert system

In philosophy, it is epistemology and ontology that constitute the study of metaphysics. Cognitive modelling might be viewed as a metaphysical study because, directly or indirectly, it models ontologies and epistemologies. Similarly, the modelling of an expert system is a metaphysical study; cognitive modelling of the expert epistemology is required.

Searle (1995, p.13), acknowledged that epistemologies might determine and process existential and social ontologies. It might be thought that naturalistic epistemologies (Kornblith, 1993) are required for existential matters and conceptual epistemologies (Harre and Madden, 1975) are required for systems of ideas. In a particular domain, these two sorts of epistemologies are

not always mutually exclusive. For instance, in the domain of law, jurisprudential systems overlay or are derived from facts, for the purpose of efficient communication between legal experts; abstracts provide this efficiency, contrary to the recommendations of Occam whose razor was used to strip science to the level of empiricism. Much of the jurisprudential system may be a matter of social ontology; for example, marriage. Law determines descriptively what actions may or may not be taken, and, prescriptively what actions must or must not be taken. The requirements for a valid contract are described; whatever defeats these requirements must not occur. It is the structure of potential processing of the legal expert's system of ideas that is the domain epistemology, and the system may contain naturalistic and conceptual matters.

Not all domain epistemologies provide the precision required for computation. Knowledge engineers have produced some generic computational epistemologies that could be thought of as products or simulations of generic human intelligence. For instance, logic is an epistemology that can derive truth from other truths; it provides truth tables that show valid arguments in the range of potential arguments. It is treated as a generic epistemology for expert systems, although a particular expertise may not be wholly monotonic in this way.

The first intelligent program, Logic Theorist (Simon and Newell, 1958), used as its computational epistemology, the heuristics or logic short cuts of logicians, that were acceptable as the common expert sense of logicians. Recently, epistemic logic programming has provided further computational epistemology resting on formal logic (Meyer and Hoek, 1995).

McCarthy (1958, 1968, 1977, 1980) built LISP, (List Processor) as a program that could store various lists of data, and process these lists in relation to each other. The standardisation of the lists and their standardised processing amounted to an epistemological template for domains for which this was adequate. The way lists were formulated and processed in LISP, determined the validity of the information produced by the program.

It could be said that the computational epistemology of some of the early knowledge base expert systems consisted of the syntax and other standardisation of the knowledge base, together with the available processes of the inference engine, usually backward or other chaining, in relation to the user's situation. Hence a formula for the computational epistemology implicit in these systems is as follows:

representation + potential processing = computational epistemology

Various expert domains that are based on practical reason with or without some logic in addition, have developed various epistemologies for various purposes. Domain epistemologies may be fragmented; sometimes they have sub-epistemologies. Generic epistemologies used by knowledge engineers may not be adequate for domains with specialised epistemologies that have idiosyncrasies; they require modifications and additions that produce a mosaic program, developed ad hoc.

Idiosyncratic expert epistemologies, such as law, may be difficult to identify. They are often the expert's trade secrets that have never been named generically or specifically, so as to keep them inherently covert. Yet there is common expert sense about what the domain epistemology is and how it works. It is the difficulty of specifying idiosyncratic domain epistemologies, in order to transform them to computational epistemologies that has produced Feigenbaum's bottleneck. As Quinlan (1979, p.168) observed:

Part of the bottleneck is perhaps due to the fact that the expert is called upon to perform tasks that he does not ordinarily do, such as setting down a comprehensive roadmap of some subject.

However, computational epistemologies may capture and incorporate idiosyncrasies from the methods, logic, heuristics and meta-rules of the domain epistemology. A domain specific shell may be designed to accommodate domain epistemology that is common to all the expert applications for which it is suited: for this purpose, common computational domain epistemology must be derived from domain epistemology, with its generic idiosyncrasies. Further epistemology is also required when the shell is used to build a particular application: the application epistemology or substantive domain epistemology that may also have idiosyncrasies.

Whether or not a computational epistemology in an expert system is adequate for a domain, depends on (1) the validity of the results it produces, and (2) whether it can produce results in the full range of the expertise sought by the intended possible user. The computational epistem-ology requires an expert's truth table. In addition, there may be user requirements that have to be met, such as cost-effectiveness that *ipso facto* includes user-friendliness; the program epistemology that allows the user to obtain new information from existing information, may vary the computational epistemology. Software engineering as well as intelligent computation may be required, and these two aspects of the system have to be integrated as program epistemology. Programming epist-emology will further transform the program epistemology to code, in accordance with the constraints of the prog-ramming software. The formalisations and schematisation of the knowledge representation, the standardised potential processing, the functionality and design of the system must all be ultimately adequate. They are all matters of meta-epistemology.

Expert system development for idiosyncratic domains may proceed systematically by a meta-epistemological method that may have five stages:

Stage 1 Domain epistemology

Stage 2 Computational epistemology

Stage 3 Shell epistemology

Stage 4 Programming epistemology

Stage 5 Application epistemology

Development proceeds by a metamorphosis of the epistemological artefact that is produced in Stage 1, in subsequent stages. Transformation of the artefact is produced by a process of reasoning that Aristotle called retroduction, currently known as abduction. Peirce (1931, p.28) observed that the Greek word for retroduction, used by Aristotle, was misinterpreted in subsequent translation as abduction.

In retroduction, a model may be posed *pro tem*, until further constraints occur to modify it. In the meta-epistemology, some constraints that arise in Stage 1 may not be relevant only in Stage 2. They may or may not become relevant in later Stages. As they become relevant, they dictate subsequent transformations. For instance, there may be a weaving of some parts of Stage 1 epistemology into Stage 2 epistemology; other parts of Stage 1 epistemology skip over Stage 2 into Stage 3 epistemology. In order to implement the Stage 3 constraints, it may be necessary to look back to the epistemology of Stage 1 to provide for constraints of Stage 1 that were not accommodated in the Stage 2 development, that can now be accommodated in Stage 3. The meta-epistemology can be thought of as a weaving method of development, akin to a lattice.

Normal text should be in style "normal", which should not be altered. It is 10 point, Times New Roman in two columns. Page size is A4 with 0.8in borders on all sides, two columns with 0.2 in between them. There should not be a blank line between paragraphs. If the style is being used properly, a half-line gap will be produced naturally.

Headings should use the heading styles as shown. Numbering is automatic.

2 Computational domain epistemology

The common computational domain epistemology of 3d legal logic was formulated in 1989 with the precision required for massive, complex legal expertise, following the development of various small legal expert systems, which could not be scaled up due to their limited epistemological basis. Representative samples of law were used to find their common computational epistemology that has the following major features:

A paradigm system of rules, procedures or strategies (RPS) provides for all possible and all alternative worlds in the range of the system; it constitutes alternative sets of jointly sufficient conditions, any one set of which is necessary to reach a final consequent or objective. This model is a sphere, constructed from interlocking of the formalised RPS.

RPS are formalised, using the paradigm of the conditional proposition, to produce computational regularity that clarifies the antecedents and consequents that are common to different RPS. These common points indicate an overlapping of RPS, where the RPS can be locked together to form the spherical model.

There are two sorts of idiosyncrasies: (a) neutral antecedents or *pseudo* sufficient conditions and (b) temporary negatives and uncertains that are *pro tem* sufficient conditions.

RPS are distinguished from commentary about these RPS. Classification of commentary allows any class to have its own sub-epistemology. These are referred to as logic *strata* that can be used to gloss any antecedent in the spherical structure.

RPS are applied to a user's case by way of extended deduction.

Figures 1-3 show graphically the construction of the spherical knowledge model. In Figure 1, rules are listed severally, as typical of a domain epistemology such as law. In the computational domain epistemology, the graphical list of rules are seen as a sequence of rivers; the rules are called rivers because they have a 'downstream' flow from one antecedent to the next until the consequent is reached. Flow structures represent potential processes.

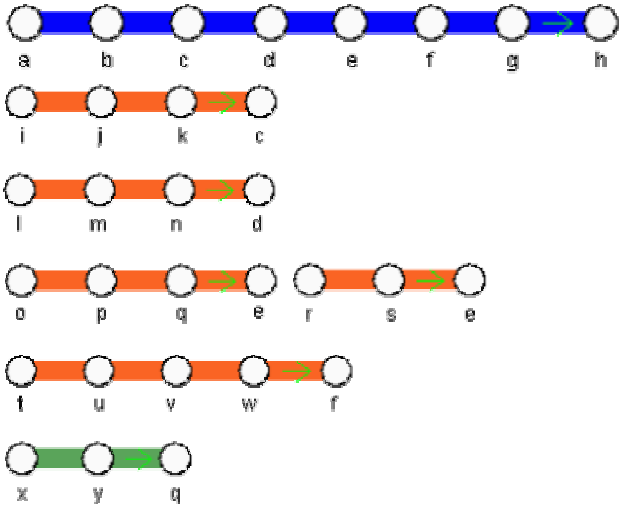

Figure 1: Rivers

In each river, all nodes, except the last, are antecedents in conjunction; the last node is the consequent. The flow is shown by an arrow that stands for 'then'. The antecedent c in the main river, that comes first, is also the consequent of the secondary river that is listed second; similarly, d, e and f are antecedents in the main river and consequents in other secondary rivers. The last river, which is a tertiary river, particularises q, which is an antecedent in the third listed secondary river. River systems are streamlined hierarchies with some common points.

Interlocking the seven rivers at common points, where they overlap, the tributary structure in Figure 2 is produced. The resulting structure allows each node to be unique. The two secondary rules that share the common consequent e can be understood as a Boolean fan; in a fan, rivers may be mutually exclusive or non-mutually exclusive. Different structures distinguish the 'and' and the 'or' aspects of RPS logic. The more upstreams for an antecedent, the more particularised or abstract it is. Many fields of law have extensive, complex river systems.

Flow of interlocked rules is always downstream; this downstream flow characterises the river flowchart. However, navigation or chaining may still go upstream or

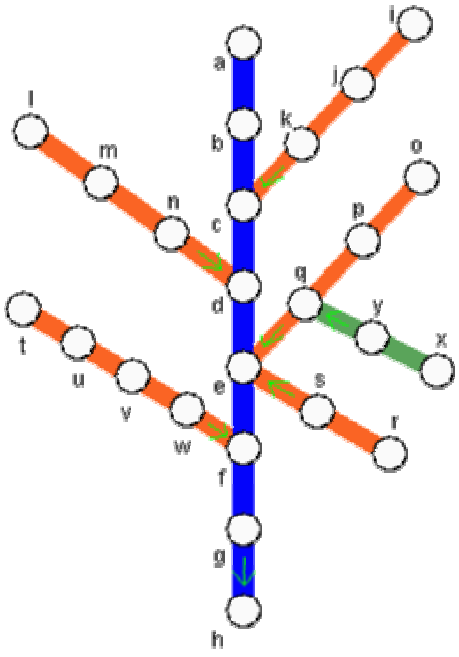

Figure 2: eGanges River Map

downstream. The river system reflects the chaining structure implicit in extended deductive arguments.

Wherever an information river is too large or complex, parts of it may be nested to control its imploding detail. This produces a system of sub-maps. The river formalisation is a reification that can be used for object-oriented programming. River ideographs resemble fishbone diagrams (Ishikawa, 1985, pp.47, 63-4, 203) that were developed in Japan in 1952 as cause and effect diagrams to assist quality and process analysis for quality control management in manufacturing. Functionality can be determined by reference to these objects.

Possible situations that fall within the expert scheme of antecedents, require corresponding positive, negative and uncertain antecedents. Whatever is stated expressly as positive, such as 'a', can be restated in its corresponding negative and uncertain forms, 'not a', and 'uncertain a'. Legal experts deal with uncertainties in their client's cases. Thus there are three corresponding rivers: the positive, the negative and the uncertain, that are linked by triad spectra at each point of corresponding antecedents. It might be thought that a positive is descriptive and a negative is prescriptive. Thus an offer, acceptance, consideration, and so on, establish a valid contract, whereas no offer precludes the establishment of a contract.

In law, some antecedents are not necessary or sufficient at all. They may be regarded as neutral antecedents or pseudo sufficient conditions that are never necessary. The meta-rule of the domain epistemology that must be picked up retroductively and accommodated for processing, is that whether or not they exist and whether or not there is certainty that they exist, is irrelevant to the final consequent. Thus, some factor that is an integral part of a situation is determined to be inconsequential. For instance, in contract law, an inquiry as to the meaning of an

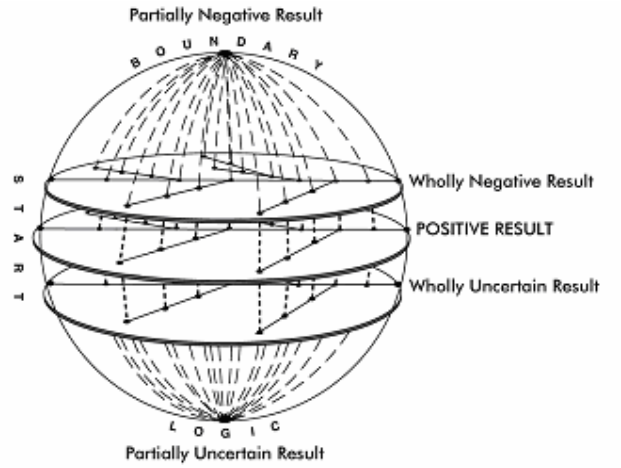

Figure 3: Spherical River Map

offer does not disband the offer. All three neutrals, enquiry, no enquiry and uncertain enquiry, are included in the positive river, without triad links to the negative or uncertain rivers. Thus positive rivers may be longer.

Before the epistemological geometry can be completed, the pole meta-rules that control combinatorial explosion must be taken into account. There are two pole meta-rules in the domain epistemology: the negative pole meta-rule and the uncertain pole meta-rule. The negative pole meta-rule provides that there are pole rules for every negative antecedent, such that if the negative antecedent is established, then there is a partially negative but conclusive Final Result. Every negative pole river shares its single antecedent with the negative hierarchical river and shares its consequent with all the other negative pole rivers; the pole is a negative fan. Establishing all the antecedents on the negative river produces the wholly negative Final Result; establishing any negative antecedent produces the partially negative but conclusive Final Result. There is a corresponding scenario for the partially uncertain pole. When pole rules are added as rivers, the final 3d logic structure is a sphere such as Figure 3.

The positive hierarchical river is placed in the equatorial cross-section as it has no pole streams. The boundary logic ring connects (1) the presupposition (*a priori*) of the system, namely that there is a first triad in the application, and (2) the postsupposition (*a posteriori*) of the system, namely that there are five alternative possible Final Results:

- partially negative but conclusive result
- wholly negative result
- positive result
- wholly uncertain result
- partially uncertain result

Notional three dimensional space is required to portray the system of RPS valid arguments. Possible cases in the legal domain occur in notional three dimensional space. Nesting of sub-maps may be seen to require a zoom into a node, like a microscope, or it might assume different layers as a depth of detail; the in-depth of sub-mapping controls combinatorial implosion in the sphere. However, the geometric structure of the triad-linked, corresponding hierarchical negative, positive and uncertain rivers, is intrinsically three dimensional; the poles cap the combinatorial explosion that occurs with possibility. This systematisation of possibility is suited to a large scale adversarial domain; it is a common universal form.

The sphere is further development in the line of symbolic geometry that represents reasoning through intellectual artefacts; this line includes (1) jurisprudential work of Fraunce (1588, 1969), Wigmore (1913, 1931), and Conover (1988), (2) the many existential graphs of Peirce (1931-6) in his nineteenth century writings, (3) logic diagrams of Venn (1894), (4) the semantic model of Korzybski (1958), and (5) the psychological maps of mind of Buzan and Buzan (1993, 1995).

Computer science has traditionally used two dimensional flowcharts and also classification and decision trees as graphical design aids. In the legal domain, for example, Capper and Susskind (1988, p.72) used an extensive decision tree in the design and construction of Latent Damage Law. Flowcharts and trees are generic (as distinct from common for possible specific applications) computational epistemology of artificial intelligence.

The epistemological sphere does map the RPS system of classification paths, interim consequents, and Final Results, as well as alternative sets of necessary and sufficient conditions. It is a form of truth table that encompasses all possible situations in the RPS system. All valid arguments are included and invalid arguments in that system are excluded. It is a specific form of flowchart or tree, arising from literal transformation of common domain epistemology.

3 Shell epistemology

The shell epistemology of eGanges (electronic glossed adversarial nested graphical expert system), allows for the implementation of the requirements of the common computational domain epistemology of 3d legal logic. Experiments carried out during the 1990s with virtual reality tools (Gray, 1995), indicated that a transformation of the sphere of rivers was required, as it was otherwise cognitively unfriendly for navigation, and too memory hungry for personal computers. The eGanges interface shown, in Figure 4, has a Rivers window where the user can Build or Consult nested RPS maps (Gray, 1988, 2002). In this Figure, the Rivers window shows the initial map of a Spam Act 2004 (Cth) application constructed by Philip Argy, partner of the law firm, Mallesons Stephen Jaques, and the author. Argy produced the first legal expert system in Australia, that was used in-house by his staff (Gray, 1997, pp.33-6).

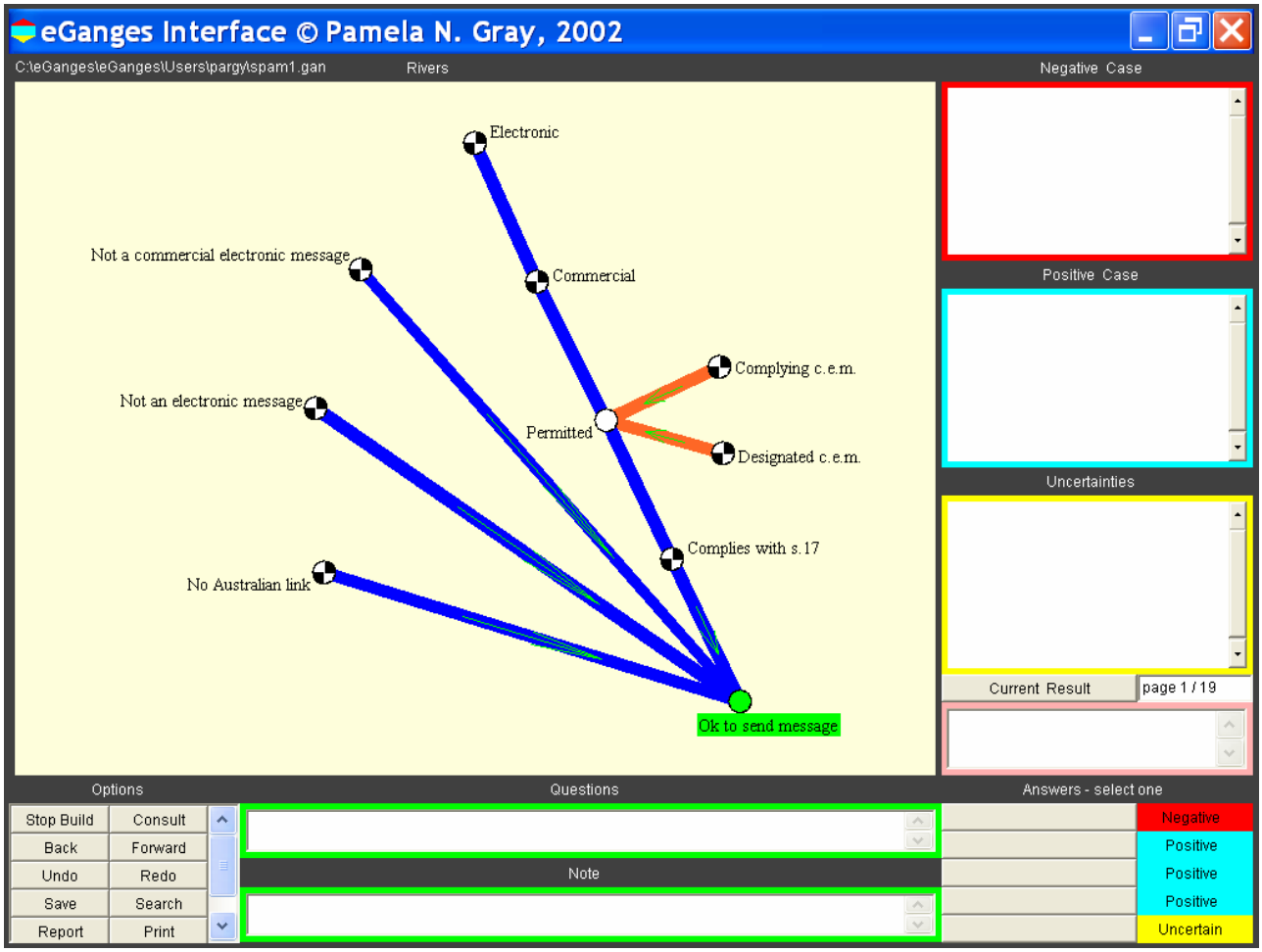

Figure 4: eGanges showing initial map of Spam Act 2004 (Cth) application (Argy and Gray, 2004)

eGanges ideographs are two dimensional positive rivers (cf. Figure 2). The shell epistemology simplifies the reification of the 3d logic model.

A node that looks like a soccer ball indicates a nested sub-map. In a sub-map, an antecedent becomes a consequent. Thus, 'Commercial' is the second antecedent in one of the four alternative initial rules that allows a message to be sent; in its sub-map, this antecedent is the consequent that is further defined by the sub-map rules.

Adversarial three dimensionality is displayed in three other windows: the Negative Case window, the Positive Case window and the Uncertainties window, where node labels are listed according to user input. This input is given as answers to questions. There is a Questions window where a user may build or consider a question that corresponds with the antecedent node that the user has selected as the current node. Beside the Questions window, there are five answer buttons, labelled for transparency: one negative, one uncertain, and, in case there are neutral nodes, three positive. The Builder may place three alternative answers to a question on three of these buttons as appropriate. For instance a yes answer may be positive or negative, and a no answer may be correspondingly negative or positive. User's can see the effect of their answers before they are given; the system is transparent for informed selection of input by the user. Immediately after an answer is given, a user sees that current antecedent label listed in a Case or Uncertainties window. At any time during a consultation, the user can see how many points there are in each list. These lists indicate user input as second premises in an extended set of *modus ponens* arguments, where the first premise is the relevant rule represented in the map.

Until all fan alternatives are exhausted, their labels will be entered in the Positive Case window, with (Neg.) or (Unc.) added as appropriate; they are consistent with the positive case. If all fans fail, as negative or uncertain, then the temporary negative or temporary uncertain labels move automatically from the Positive Case window to the appropriate Negative Case or Uncertainties window. Otherwise, to move an antecedent from one case window to another, requires the answer to be changed.

A positive label may contain factual negation, such as 'no rejection' (of an offer); when this label appears in the Negative case window it indicates, by virtue of the double negative, the fact of rejection. The Builder may warn of double negations. Below the Questions window is a Note window for any warnings or help messages about questions. These are Question glosses. The Consult user may also type input, termed User glosses, into the Note window to record user evidence, concerns etc in the Consultation report. Here the shell epistemology of eGanges meets any language diversity, epistemology or ontology of the Consult user.

Other gloss facilities are available for the Builder to add further commentary on any node. The following eGanges strata logic facilities are available for different types of glosses, according to the requirements of the domain epistemology, at their precise point of relevance (Gray and Gray, 2003):

(1) text for noting authorities or justification for rules,

(2) a spectrum of three sectors for delineating distinctions between positive, negative and uncertain antecedents,

(3) links between nodes in different rules,

(4) links to parallel river systems, or to other files or programs.

Extended legal arguments may have elements of gloss arguments about what the rules mean, why they are rules or what the rules ought to be. In the domain epistemology, gloss content may be interwoven into the presentation of arguments as a reinforcement of the RPS system, or as a justification for its modification. While node labels are short and quick, with associated questions, glosses may expand on the these labels and questions for deeper understanding. Comprehension may occur in this range, as required by the user. Because subjects of the realm are required to obey the law, a rational legal system will facilitate its own effectiveness by providing ready access to the law, and understanding of it. This is fundamental to the domain epistemology.

There is also a Current Result button and window. A label in the Negative case list produces a negative Final Result that overrides established positive and uncertain antecedents. Both the partially negative and wholly negative results are Final results. If the negative case window has nothing, a label in the Uncertainties window produces a *pro tem* current result of uncertain, pending total failure of all negatives. If there is nothing listed in the Negative case or Uncertainties windows, there is a *pro tem* current result, 'unanswered', pending failure of all negatives and uncertains when the positive Final Result prevails.

In other domains, the case with the majority of points, might win; antecedents may be weighted differently for this calculation. However, in the domain epistemology of law, all jointly sufficient antecedents in at least one positive set must be established for a positive final result; positive fans that may appear in any nest level, by combinatorial implosion, produce alternative positive sets. By virtue of the uncertainties that are not found to be positives, the negative case ultimately wins if the positive case has the burden of proof.

The eGanges interface can communicate large complex RPS systems through a common suburban street map paradigm and interactive concrete visualisation; the shell is easy for the user to understand, learn and use. Xenogene Gray (Gray and Gray, 2003) carried out the programming transformation using Java, to give effect to the shell epistemology.

4 References

Buzan, T. and Buzan, B. (1995; first edition, 1993): *The Mind Map Book.* London, BBC Books.

Capper, P. and Susskind, R.E. (1988): *Latent Damage Law - The Expert System*, London, Butterworths.

Conover, M. (1988): Applying Three-Dimensional Thinking and Systems Technologies to Jurisprudence

and Legal Management. In *Interactive Systems and Law*. Rasmussen, T. (ed.). Lansing, Michigan, Spartan Press.

Fraunce, A. (1969. Reproduced from the original manuscript in the British Museum, *Lawiers Logike* first published in 1588): *The Lawyer's Logic*. Menston, England, The Scolar Press Limited.

Gray, P.N. (2002): Explanatory Rule Maps. *Computers and Law Journal for the Australian and New Zealand Societies for Computers and the Law* 50:11-13, Dec.

Gray, P.N. (1997): *Artificial Legal Intelligence*. Aldershot, England, Dartmouth Publishing Co.

Gray, P.N. (1995): Scaling Up To A Three Dimensional Graphic Trace. In Ciampi, C., Socci Natali, F. and Taddei E.G. (eds), *Verso Un Sistema Esperto Giuridico Integrale*. Padua, Italy, Cedam.

Gray, P.N. (1990): Choice and Jurisprudential Systems. LL.M. thesis. University of Sydney, Sydney.

Gray, P.N. (1988): The CLIMS (Contract Law Information Management System) Pilot - Automatable Law. In *Proc. of the 4th International Congress on Computers and Law*, Rome..

Gray, P.N. and Gray, X. (2003): A Map-Based Expert-Friendly Shell. In *Legal Knowledge and Information Systems*. Bourcier, D. (ed.). Amsterdam, IOS Press.

Harre, R. and Madden, E.H. (1975): *Causal Powers, A theory of natural necessity*. Oxford, England, B. Blackwell.

Ishikawa, K. (1985): *What Is Total Quality Control? The Japanese Way*, translated by David J. Lu. Englewood Cliffs, N.J., Prentice-Hall Inc.

Kornblith, H., (1993): *Inductive Inference and Its Natural Ground An Essay in Naturalistic Epistemology*. Cambridge, MA, MIT Press.

Korzybski, A. (4th ed. 1958): *Science and Sanity: An Introduction to Non-Aristotelian Systems and General Semantics*. Lakeville, Connecticut, The International Non-Aristotelian Library Publishing Company.

Lifschitz, V.(ed.) (1990): *Formalizing Common sense: Papers by John McCarthy.* Norwood, N.J., Ablex Publishing Company.

McCarthy, J. (1958): Programs with common sense. In Proceedings of the Symposium on Mechanisation of Thought Processes, volume 1, pp.77-84. London, England, Her Majesty's Stationery Office.

McCarthy, J. (1968): Programs with common sense. In Minsky, M.L., (ed.), Semantic Information Processing. Cambridge, MA., MIT Press.

McCarthy, J. (1977): Epistemological problems in artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (IJCAI-77). Cambridge, MA, IJCAII.

McCarthy, J. (1980): Circumscription: a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27-39.

McCarthy, J. and Hayes, P.J. (1969): Some Philosophical Problems from the Viewpoint of Artificial Intelligence. In *Machine Intelligence 4*. Meltzer, B. and Mitchie, D. (eds). Edinburgh, Edinburgh University Press.

Meyer, J. and Hoek, W van der (1995): *Epistemic logic for AI and Computer Science*, Cambridge, Eng., Cambridge University Press.

Peirce, C.S., (1931) Principles of Philosophy, *Collected Papers* Vol.1, Cambridge, Massachusetts, Harvard University Press.

Peirce, C.S., (1931-6) *Collected Papers* Vols.1-6, Cambridge, Massachusetts, Harvard University Press.

Quinlan, J.R. (1979): Discovering Rules by Induction from Large Collections of Examples. In *Expert Systems in the Micro-Electronic Age*. Michie, D.E. (ed.). Edinburgh, Edinburgh University Press.

Samuel, G. (2003): *Epistemology and method in law*, Burlington, VT, Ashgate.

Searle, J.R. (1995): *The Construction of Social Reality*. Harmondsworth, Middlesex, Penguin.

Simon, H.A. and Newell, A.(1958): Heuristic problem solving: The next advance in operations research. *Operations Research*, 6:1-10.

Venn, J. (2nd ed.,1894): *Symbolic Logic.* London.

Wigmore, J.H. (1913, 1931): *Principles of judicial proof*. Boston, Little, Brown and Company.