Controlling and augmenting legal inferencing: *ysh*, a case study

Graham Greenleaf, Faculty of Law, University of New South Wales
(graham@datalex.law.uts.edu.au)
and
Andrew Mowbray, Faculty of Law and Legal Practice, University of Technology, Sydney
(andrew@datalex.law.uts.edu.au)

Abstract: If legal inferencing systems are to be used for immediate practical application, they are best constructed by embedding them in other technologies which can assist in augmenting and controlling the course of inferencing. Adoption of a (quasi) natural language knowledge representation assists easier development of user interpretative facilities, user control of the course of inferencing and explanation facilities. The paper explains how the *DataLex Workstation Software*, particularly its inference engine, *ysh*, implements these approaches.

Augmenting and controlling inferencing

To develop legal inferencing systems in isolation from other technologies for representing and manipulating legal information is unlikely to yield systems of immediate practical application. There are two broad reasons for this assertion.

Augmenting inferencing

The first reason arises from the inherently and endemically open textured nature of legislation and case law. The lack of fixed meanings of the language of cases and statutes means that it is not possible, even in theory, for an application developer to anticipate all possible factual circumstances which may come within the meaning of predicates used in system dialogues. Inferencing techniques to resolve unanticipated open texture problems are not yet commercially viable[1].

A legal inferencing system must therefore constantly require the user to make significant interpretative decisions. This makes it necessary to give the user effective and open-ended access to a wide variety of textual interpretative materials. The overall goal is to build an inferencing system that most effectively supports the user's interpretative activity, allowing control of a problem's solution to alternate between a semi-expert system and a semi-expert interpretive agent, the user (Greenleaf, Mowbray and Tyree, 1991).

Controlling inferencing

The second, related, problem is that the depth or extent to which a user will want, or need, to use aspects of a legal inferencing system to solve problems will vary. This may depend upon either the extent of the user's domain expertise, the facts of the problem at hand, or both. For example, in an inferencing system on Australian privacy law, consultations concerning compliance with the Information Privacy Principles[2] may depend upon whether the alleged breach is by an 'agency'. 'Agency' has a complex definition[3] which would often result in a lengthy inferencing dialogue. Most likely users could answer without hesitation that the Department of Social Security was an 'agency' (ie usage is problem dependent). However, only experienced users would unhesitatingly answer that the Commonwealth Ombudsman was an 'agency' - most would want to use any available inferencing facilities (ie usage is expertise dependent). In other cases, effective resort to textual materials will be far more efficient than running an inferencing session. For example, most users would not immediately know whether the Supreme Court of the Australian Capital Territory is an 'agency', but quick reference to the s6 definition of 'agency' will make the answer apparent .

If users are always forced to pursue tedious inferencing dialogues to the bitter end they are unlikely to find inferencing systems efficient, attractive to use, or 'intelligent'. A system which insists upon dealing with every minor concern at the greatest level of detail is unlikely to be acceptable. On the other hand, users cannot just be left to their own resources to decide how to resolve questions asked of them in inferencing dialogues, but need precise prompts as to the resources they can access (further inferencing, access to commentary, statutes etc).

This second problem, user control of depth of inferencing, is an instance of a broader principle that, for a knowledge-based system to be of the greatest practical utility, the user needs to be able to exercise various types of control over its operation. The initial invocation of a consultation must be appropriate to a problem at hand. The open-ended nature of interpretative problems means that a user who is attempting to resolve a question asked by an inferencing dialogue may need to run other inferencing sessions unconnected to the primary session as, in effect, 'sub-consultations', and then be able to return to the original consultation. To achieve this degree of user control, a wide variety of access mechanisms to inferencing are needed.

This paper explores two aspects of the DataLex Workstation Software which attempt to deal with these problems: user control of inferencing, and the advantages of a (quasi) natural language knowledge representation.

DataLex Workstation Software

The DataLex Workstation Software[4] integrates inferencing systems, hypertext and text retrieval into a general-purpose tool for the representation and processing of legal information. Its origins are outlined in Greenleaf, Mowbray and Tyree (1991), with a number of features foreshadowed there since completed. Its features are documented in Greenleaf and Mowbray (1992) and (1993). It now runs under *Microsoft*

Windows, Apple Macintosh, Unix and *MS-DOS*. It has been used to develop several commercial and academic applications, and has been used extensively for teaching purposes[5].

The Workstation's inference engine

The inference engine used by the DataLex Workstation software is called *ysh* (we pronounce it 'why-shell'). *ysh* supports rule-based and case-based inferencing, and document generation, using rules which have a similar syntax. Only rule-based inferencing is discussed here. By default, rules are both backward and forward chaining. *ysh*'s inferencing mechanism works by evaluating a rule which has been selected by the user (either from a list of goals, or from a cross-reference index). Inferencing proceeds by the execution of statements comprising a rule. Backward chaining is used to determine unknown facts. When a fact value is inferred or supplied by the user, evaluation of all rules which contain that fact in their conditions takes place by forward chaining.

Rules used in rule-based reasoning may be declared to be of types FORWARD, DAEMON, or PROCEDURE. A FORWARD rule will ask for user intervention (if necessary) to provide the unknown values of conditions, whereas a DAEMON operates 'silently', failing rather than asking for user intervention. A PROCEDURE must either be explicitly called by another rule or explicitly invoked by the user . Any rule may be declared to be subject to two rule qualifiers. GOALs are not automatically when invoked by backward chaining — user confirmation is required before inferencing proceeds. Titles of LISTED rules are displayed in *ysh*'s menu of goals. GOAL rules are, by default, also LISTED, and are intended to be those rules appropriate to commence a significant separate inferencing process.

ysh's knowledge representation

We call *ysh*'s knowledge representation 'quasi natural language' for two reasons. At least where statutory materials are concerned, it often approximates a paraphrase of the legislation. Secondly, it permits re-parsing of these English-like rules to generate inferencing dialogues. Similar approaches have been taken by Waterman[6] and by Johnson and Mead[7].

Facts

All dynamic information is stored as a set of *facts*. Facts are referred to by a descriptive phrase or sentence. Most facts are propositional and may have one of four values: true, false, unknown, or unknowable. The name of this type of fact should be composed of a subject, then a verb (expressed in the positive or negative) and, optionally, an object. For example:

```
the intestate satisfies s23(1)
the work is "original"
section 9 applies
```

Provided that boolean fact names appear in this form, *ysh* will normally be able to affect sensible translations (by a simple set of heuristic rules) which can be used during problem sessions. For the first example above, the following automatic prompts and translations would be used:

```
Does the intestate satisfy s23(1)?
The intestate satisfies s23(1).
The intestate does not satisfy s23(1).
```

Apart from propositions facts can also be used to refer to numbers, amounts, dates and gender.

Rule syntax

Rules have a straight forward syntax. The following example shows s135ZK of the *Copyright Act 1968* (Cth), and the first three rules representing it.

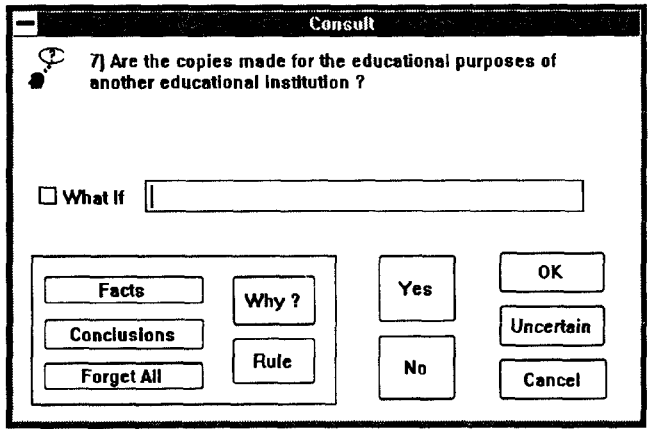
```
Multiple copying of works published in anthologies
135ZK. The copyright in a literary or dramatic work,
being a work contained in a published anthology of works
and comprising not more than 15 pages in that anthology,
is not infringed by the making of one or more copies of
the whole or a part of that work by, or on behalf of, a
body administering an educational institution if:
     (a) a remuneration notice, given by or on behalf of
the body to the relevant collecting society, is in
force;
     (b) the copy is made solely for the educational
purposes of the institution or of another educational
institution; and
     (c) the body complies with subsection 135ZX (1) or
(3), as the case requires, in relation to the copy

RULE Copyright Act 1968 - Section 135ZK PROVIDES
section 135ZK applies ONLY IF
  the work is a literary work AND/OR the work is a
  dramatic work AND
  the work is contained in a published anthology of works
  AND
  the work does not comprise more then 15 pages in the
  anthology AND
  the work has been copied by the body administering the
  educational institution AND/OR the work has been copied
  on behalf of the body administering the educational
  institution AND
  section 135ZK(a) applies AND
  section 135ZK(b) applies AND
  section 135ZK(c) applies

RULE Copyright Act 1968 - Section 135ZK(a) PROVIDES
section 135ZK(a) applies ONLY IF
  a remuneration notice given by or on behalf of the
  relevant collecting society is in force

RULE Copyright Act 1968 - Section 135ZK(b) PROVIDES
section 135ZK(b) applies ONLY IF
  the copies are made for the educational purposes of the
  educational institution OR
  the copies are made for the educational purposes of
  another educational institution
```

Isomorphism has been retained to a substantial degree by explicit modelling of the structure of the section (Johnson and Mead, 1991, p113).

Explanation facilities in *ysh*

ysh provides a range of explanations of its inferencing (see Greenleaf, Mowbray and Tyree, 1991, at 6.3). All of these explanations (except the 'rule' option) are generated by *ysh* reparsing parts of the rule-base on the fly. These explanations do not, therefore, go beyond the content of the rule base. The reparsing is affected by a simple set of heuristic rules.

Inferencing dialogues are initiated principally through the window illustrated.



For examples of this reparsing, based on the above rules representing s135ZK, assume that earlier user responses in a consultation satisfy the introductory conditions of the head section itself and 135ZK(a), but not the first half of 135ZK(b). The question (prompt) 'Are the copies made for the *educational purposes* of another *educational institution*?' will then be generated by the re-parsing of the second condition of the rule for 135ZK(b). If the user selects the 'Why?' button when faced with this prompt, the explanation 'This will help determine whether or not *section 135ZK*(b) applies'. If the user's answer is 'no', this fact would be translated as 'the copies are not made for the *educational purposes* of another *educational institution*'. The conclusions inferred at this point would be translated as '*section 135ZK*(b) does not apply' and '*section 135ZK* does not apply'. If the user asks for an explanation of the second of these conclusions, the explanation generated would be '*Section 135ZK*(b) does not apply because the copies are not made for the *educational purposes* of the *educational institution* and the copies are not made for the *educational purposes* of another *educational institution*'. On completion of the consultation, *ysh* generates a report based upon the dependencies between premises and conclusions established by inferencing, such as: '*Section 135ZK* does not apply because *section 135ZK*(b) does not apply. *Section 135ZK*(b) does not apply because the copies are not made for the *educational purposes* of UTS and the copies are not made for the *educational purposes* of another *educational institution.*'

A complementary general feature of these explanations is that they are integrated with the hypertext aspects of the system, so that most explanations appear as hypertext, allowing the user to go from the terse explanations generated by *ysh* to more expansive source documents and explanatory text. All of the italicised words in the above dialogues appear as hypertext.

Integration of *ysh* with hypertext and text retrieval

There are six types of integration between the inferencing, hypertext and search engines implemented in the Workstation software (Greenleaf, Mowbray and Tyree (1991) at 7.1). Consultations using the knowledge-base are usually started from within the hypertext portion of the system. For example, if the user is reading some part of a section of an Act or a defined term, a consultation dealing with this material and applying it to a specific problem faced by the user can be started by selecting an inferencing session from the cross-reference index (shown below) for that text. The hypertext acts as a distributed interface to the knowledge-base. Consultations may also be commenced from the list of GOALS.

Once the consultation is underway, the user can go back into the hypertext system via selection of hyperterms embedded in the questions and explanations produced by the inferencing process. Once in the textual materials, the user can again resort to the knowledge-base if necessary, by commencing other consultations from the cross-reference index.

Advantages of a (quasi) natural language knowledge representation

A number of advantages flow from the choice of this type of knowledge representation. Some of these advantages have been shown to flow from isomorphic knowledge representations (Bench-Capon and Coenen (1991) and Johnson and Mead (1991)), but we argue that these benefits can be enhanced by the type of knowledge representation used in *ysh*, and other benefits obtained.

The source of some of these additional benefits is the automatic generation of all aspects of inferencing dialogues by the reparsing of the knowledge-base. As explained, this includes the generation of prompts, translations of user-supplied and system-inferred fact/values ('Fact' and 'Conclusion'), explanations of questions ('Why') and conclusions ('How'), and construction of a consolidated Report.

Isomorphism is facilitated

The advantages of maximising the isomorphism of the relationship between a knowledge-base and the sources of law on which it is based have been advanced by Bench-Capon and Forder (1991), Bench-Capon and Coenen (1991) and Johnson and Mead (1991). Criticisms of isomorphism by Moles (1991) confuse questions of its desirability and possibility with criticisms of the particular features of representations used by Bench-Capon et al, which involved an intermediate representation which is not a necessary feature of other systems. They also seem to be based on the unrealistic assumption that developers of legal inferencing systems expect their systems to be comprehensive in ways that no-one expects of those who package legal expertise in other forms such as textbooks. As Tyree (1992) says, it is hard to dismiss isomorphism as a desirable goal unless one's approach is an *a priori* 'give up'.

A rule-based knowledge representation which is in an English-like propositional form, such as *ysh* or the representation used by Mead and Johnson in STATUTE, can have a high degree of isomorphism, both structurally (ie reflecting the logical structure of sections and subsections) and in much of the actual language used. Such isomorphism still, of course, falls well short of a knowledge representation which is identical with natural language[8], partly because of the limitations of the logical operators which have yet been developed even in logic programming (Routen, 1989), and ultimately because the choice of operators is still an interpretation (Allen and Saxon, 1991).

Isomorphism is also facilitated by the almost complete elimination of prompts, translations, explanations etc being

represented separately from rules. To the extent that any of this 'textual baggage' (Johnson and Mead, 1989) is required by a knowledge representation, there is a loss of isomorphism.

Increase in transparency of knowledge-base

A (quasi) natural language knowledge-base provides advantages of transparency to the application developer, to domain experts and to the user. Both the application developer and any domain experts can more easily make visual comparisons between the knowledge-base and the legal sources from which it is derived if the knowledge representation is close to English and attempts 'verbatim modelling' of statutory predicates as far as possible (Johnson and Mead, 1991, p112). This enhances the validation advantages of isomorphism (Bench-Capon and Coenen, 1991).

Providing the user of the application with open ended access to the knowledge-base during the operation of the system (as is done through the 'Rule' command and hypertext browsing) is desirable for several reasons. Most obviously, it allows the user to check for completeness and accuracy. A transparent knowledge base helps to demystify and explain exactly what the system can and cannot do. It assists with user confidence in the results generated by problem sessions and discourages blind reliance. This depends upon the knowledge-base being intelligible to likely users.

Increased explanatory power

The integration of the inferencing component with hypertext and text retrieval increases the explanatory power of what are otherwise necessarily terse explanations because they are automatically generated from the rule-base.

Efficiency in application development

An application developer does not have to give any significant amount of attention to the development of the 'textual baggage' involved in prompts, translations, explanations etc. If the application developer can concentrate solely on writing a rule base, this should lead to more efficient application development. If the rule base can be made as close as possible to a paraphrase of the legislation (or other source), this should also cause more efficient development. Bench-Capon and Coenen (1991) have found that 'adopting the principles of isomorphism results in a very disciplined and teachable methodology', but this may apply even more strongly to knowledge representations which are also English-like rather than symbolic (Johnson and Mead, 1991).

Reduced maintenance of knowledge-base

Ease of maintenance, a further advantage of isomorphism (Bench Capon and Coenen, 1991, p67), will be enhanced for the same reasons. If only one source requires updating, there is less likelihood of errors and more opportunity for efficiency.

Control of Inferencing in *ysh*

Developer control of inferencing behaviour

By default, all *ysh* rules are both backward chaining and forward chaining daemons, usually written in a purely declarative fashion. However, an application developer may decide to give consideration to how rules will behave by declaring rules to be of various types (PROCEDURE, DAEMON, or FORWARD), or as having qualifiers (GOAL), thereby modifying the normal course of inferencing. This may be done in order to create more efficient inferencing in the particular application, in which case it represents the embodiment of application-specific heuristics. It may also be done in order to allow the user (in a specific application) to assume greater control over the course of inferencing than the default behaviour would allow.

In either case, the application developer is, in effect, imposing a level of procedural control over an otherwise declarative rule-base and non-procedural inferencing system. This could be considered to detract from the declarative nature of the rule-base, and also from its isomorphism. However, the declarative content of the rules is unaffected by this approach, so we argue that it is a reasonable compromise.

We have two general guidelines for the use of rule types. First, it is best to first write the rule base with default rules only, then observe its behaviour and only modify it to the minimum extent necessary. Second, the use of heuristic rules and rules which include any procedural elements should be minimised, and they should be kept separate from any other rules.

Some examples of use of various rule types follow. DAEMON is the only rule type which has any frequent use, particularly in relation to heuristic rules. For example, in a propositional knowledge representation, it may be useful to have a heuristic rule that provides that if a work is a literary work it is not an artistic work. However, if the system needs to determine whether a work is an artistic work, then it would be pointless for it to back-chain to this rule and ask whether the work is a literary work. The use of DAEMON turns the back chaining off.

FORWARD rules are rarely used but can be valuable. They are, in effect, heuristics to increase the inferencing efficiency by embodying known 'short cuts'. For example, many laws concerning government agencies only apply to the Australian Security Intelligence Organisation (ASIO) in very restricted circumstances. If the system determines at any stage that it is dealing with ASIO, it may be valuable to have a FORWARD rule which is triggered and interrupts the normal chain of inferencing in order to determine the values of the other conditions of 'the ASIO rule', even if this involves asking the user to provide fact values.

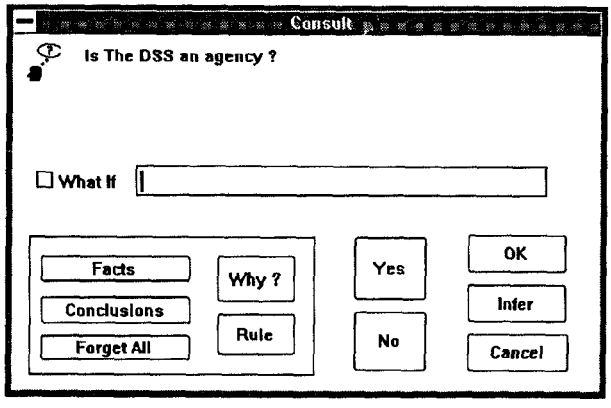
Other minor features allow elements of control. In backward chaining, where two or more rules have as conclusions the fact that is being evaluated, *ysh* evaluates these rules in the order in which they appear in the knowledge-base. If this order needs to be changed, and ORDER statement can be used without changing the isomorphic nature of the representation. Another feature of this nature is the use of the boolean non-conditional operators OR/WITH and AND/WITH, which cause *ysh* to determine all grounds on which a conclusion is supported, not just sufficient grounds to justify it. Their use changes the inferencing behaviour, but only in the sense of prolonging it.

User control of extent of inferencing

The main use of rule types is to allow the application developer to limit the depth of backward chaining which occurs automatically. Wherever *ysh* is evaluating a fact, a value for which can be inferred by back chaining to a rule which is a declared to be a GOAL, *ysh* will require the user to confirm that further chaining is to occur, and will give the user the option of providing a value for the fact instead. In effect, control of the depth of backward chaining is passed to the user.

This is illustrated by the Consult window below. In contrast with the normal Consult window, the 'Uncertain' button has been replaced by an 'Infer' button, to indicate that further inferences are available to determine a value for the fact 'the

DSS is an agency'. The hypertext highlighting of the word 'agency' also indicates to the user that a definition of the term 'agency' can be accessed by selecting that hypertext link. The user is therefore given three choices: (i) to answer the question without reference to any form of assistance; (ii) to pursue hypertext links to textual materials before answering; or (iii) to use further inferencing to resolve the question. In this way, *ysh* caters for differences in user expertise, and for differences in the difficulty of the problem at hand.



It is up to the application developer to decide the extent to which backward chaining should be automatic. However, our general view is that the depth of inferencing should normally be left to the user, at least in relation to those rules which could be considered to commence the evaluation of significant goals in themselves (hence the name 'GOAL' for this rule type).

Conclusions

Our experience in the development and use of the DataLex Workstation Software has been that the integration of inferencing systems with other technologies for representing and processing legal data and knowledge continues to give new benefits in the development, control and utility of the inferencing components. The use of a (quasi) natural language knowledge representation makes this approach far more effective. These features also provide an appropriate context for further development of case-based reasoning and document generation.

References

Allen and Saxon (1991) L Allen and C Saxon 'More IA needed in AI: Interpretation assistance for coping with the problem of multiple structural interpretations' *Proc. 3rd ICAIL* ACM Press 1991

Bench-Capon and Forder (1991) T Bench-Capon and J Forder 'Knowledge representation for legal applications' in Bench-Capon (1991)

Bench-Capon and Coenen (1991) T Bench-Capon and F Coenen 'Exploiting isomorphism: Development of a KBS to support British Coal insurance claims' *Proc. 3rd ICAIL* ACM Press 1991

Brown (1993) G Brown 'CHINATAX: Exploring isomorphism with Chinese law' (Proc. 4ICAIL), 1993

Bubna-Litic (1993) K Bubna-Litic 'Corplaw1: A legal expert system to teach company law' (unpublished 1993)

Greenleaf and Mowbray (1992) *DataLex Workstation Software - Application Developer's Manual* , DataLex Pty Ltd, 1992

Greenleaf and Mowbray (1993) *DataLex Workstation Software - Interim User Guide* (Windows version 3.0.1), DataLex Pty Ltd, 1993

Greenleaf, Mowbray and Tyree (1991) 'The DataLex Legal Workstation', *Proc. 3rd ICAIL* ACM Press 1991

Greenleaf, Mowbray and Tyree (1992) 'The Privacy Workstation' 6 *Yearbook of Law, Computers and Technology*, 1992

Johnson and Mead (1989) P Johnson and D Mead 'Legislative expert systems' and 'Natural language - An appropriate knowledge representation scheme for legislative expert systems' (unpublished), SoftLaw, Canberra, 1989

Johnson and Mead (1991) P Johnson and D Mead 'Legislative Knowledge Base Systems for Public Administration' *Proc. 3rd ICAIL* ACM Press 1991

Mital and Johnson (1992) V Mital and L Johnson *Advanced Information Systems for Lawyers* Chapman & Hall, 1992

Moles (1991) R Moles 'Logic programming - An assessment of its potential for artificial intelligence' *Journal of law and Information Science* Vol 2, 1991

Routen (1989) T Routen 'Hierarchically organised formalisations' *Proc. 2nd ICAIL* ACM Press 1989

Sergot (1991) M Sergot 'The representation of law in computer programs' in Bench-Capon (1991)

Tyree (1992) A Tyree 'The logic programming debate' *Journal of Law and Information Science* Vol 3 No 1, 1992

Wahlgren (1992) P Wahlgren *Automation of Legal Reasoning* Kluwer, 1992

Waterman et al (1986) D Waterman, J Paul and M Peterson 'Expert systems for legal decision making' *Proc. 2nd Aust. Conf. on Expert Systems* NSW Institute of Technology, 1986

Waterman and Peterson (1986) D Waterman and M Peterson 'Models of legal decision making' in P Clahr and D Waterman (Eds) *Expert Systems - Techniques, tools and Applications* Addison Wesley 1986 Ch 5

Notes

[1] Despite important and valuable research by Ashley, Gardner and many others: see Mital and Johnson (1992) Chapter 14, Sergot (1991) 1.17 and Wahlgren 6.4.5.2 for summaries.

[2] *Privacy Act 1988* (Cth) s14; Implemented in the Privacy Workstation: see Greenleaf, Mowbray and Tyree (1992).

[3] *Privacy Act 1988* (Cth) s6

[4] © DataLex Pty Ltd; designed by A Mowbray and G Greenleaf; written by A Mowbray

[5] Applications of the DataLex Software include those dealing with privacy and data protection law (Greenleaf, Mowbray & Tyree, 1992)), intellectual property law, Chinese tax law (Brown, 1993) and corporations law (Bubna-Litic, 1993). It has been used to develop over fifty student applications and has been licensed to law schools in Australia and internationally.

[6] See the work of D Waterman et al on ROSIE; for example Waterman et al (1986) and Waterman and Peterson (1986), Ch 5

[7] developers of the STATUTE software; see Johnson and Mead (1989) and (1991)

[8] See Sergot (1991) 1.4.5.: '...no project to my knowledge has attempted to reason directly with the sources of law.'