Law as Code:
Introducing AustLII's DataLex AI

Andrew Mowbray, Graham Greenleaf and Philip Chung*

16 November 2021 – Draft only, submitted for publication in *Computers & Law* (AUSCL)

AustLII,[1] as a provider of free access to legal information, has a distinctive approach to the use of artificial intelligence (AI) in relation to legal information. This article introduces that approach, which is called 'DataLex'.[2]

'Rules as Code' includes the activity of converting a legal text which is in a natural language (legislation, regulations, or other legal instruments – generically, 'law' or 'rules'), into a representation in a computer-processable language (code). One application of this is that when processed by an appropriate program, this allows a human user to input data on a particular fact situation, and thereby produce conclusions which are an accurate statement of the legislative intent of the legal text when applied to that data. The current shorthand is 'rules as code' or 'law as code'.

Objectives and constraints

The DataLex approach is based on symbolic representation and processing of information, particularly of sets of rules. In contrast, most advances in artificial intelligence in recent decades have been based on machine learning (ML), particularly in such areas as image recognition or language translation. ML has had limited success in relation to legal information.

Legislative rules can be seen as being prescriptive, regulating social, business and economic activities. The temptation is to simply model these prescriptions, but this leads to an immediate conceptual fork between what legislative rules 'say' and what they are taken to 'do'. The approach AustLII recommends is to avoid trying to represent what rules do, and instead to represent what they say. We regard legislative rules as a hierarchical set of propositions which include both real world references as well as propositions about how legislation works. This has advantages both in scaling up the development of code-bases, and in maintaining their isomorphism with the legislation on which they are based.

AustLII is part of the free access to law movement, and therefore has a particular interest in how AI may be used to create valuable extensions of free access to legal information. One

---

* Andrew Mowbray is Professor of Law & Information Technology at UTS and Co-Director of AustLII; Graham Greenleaf is Professor of Law & Information Systems at UNSW and Co-Founder of AustLII; Philip Chung is Associate Professor of Law at UNSW and Executive Director of AustLII.

[1] *Australasian Legal Information Institute* (AustLII) <https://www.austlii.edu.au> a joint facility of the Faculties of Law at UNSW and UTS. AustLII, founded in 1995, is the largest provider of free access to legal information in Australia, with over 700,000 page accesses per day to its Australasian site.

[2] The DataLex website is at < http://www.datalex.org/> .

aspect of this is to ask how can the creation and maintenance of legal code-bases be made simple and affordable enough so that the legal assistance sector (and not only the commercial sector) can be involved in it. We are interested in building and maintaining legal code-bases on a large scale and in doing so in a cost effective way. Can legal code-bases be made accessible for public use in the same way that AustLII has made legal databases available?

The DataLex approach therefore involves the use of a less common approach to AI (symbolic representation), it aims to represent what rules say, not what they do, and it works within the constraint of assuming that resources available for AI developments will be limited.

Technical components: The DataLex toolkit

The following technical components (the 'DataLex toolkit') contribute to DataLex's overall effectiveness. We explain them in more detail later, where necessary. The diagram following represents the relationships between these components.

1. **The *yscript* language** – A language for representing and manipulating propositions, yscript can be used to represent real-world rules such as legislation, codes of conduct and other sets of rules. The yscript language is at the heart of the DataLex approach.[3] *Coding with yscript* (May 2021) is a complete description and tutorial introduction to coding in the yscript language. *AustLII's DataLex Developer's Manual* (June 2021) introduces new developers to writing DataLex applications. It explains the basic concepts and includes step-by-step instructions on how to use yscript and the DataLex Development Tools (5 below) to build and run applications.
2. **The *yscript* interpreter** – This interpreter program runs or processes sets of rules expressed in the yscript language (provided they are syntactically correct) in order to engage a user in an interactive consultation. It produces a report on conclusions generated during the consultation, or other outputs such as documents. The yscript interpreter is available under a GPL licence.[4]
3. **The *ylegis* pre-processor program** – The *ylegis* pre-processor converts the text of legislation into *yscript* rules that reflect the legislation's structure. The resulting pre-processed yscript code can immediately be run by the yscript interpreter, to test (via observing consultations, and by use of error-checking tools) how well the conversion to code has worked. To the extent that this is successful,[5] it is the automated conversion of 'rules into code' or 'law into code'. Until ylegis (2021), development of the *yscript* code was an entirely manual process, aided only by some error-checking tools. Technically, ylegis can be described as either a filter, or as a pre-processor. A short paper *The DataLex legislation pre-processor for rules as code* includes examples.[6]
4. **Formal mode of ylegis** – In its formal mode, ylegis can be used to write *y*script code more conveniently and succinctly using a formal set of connectors (operators) such as

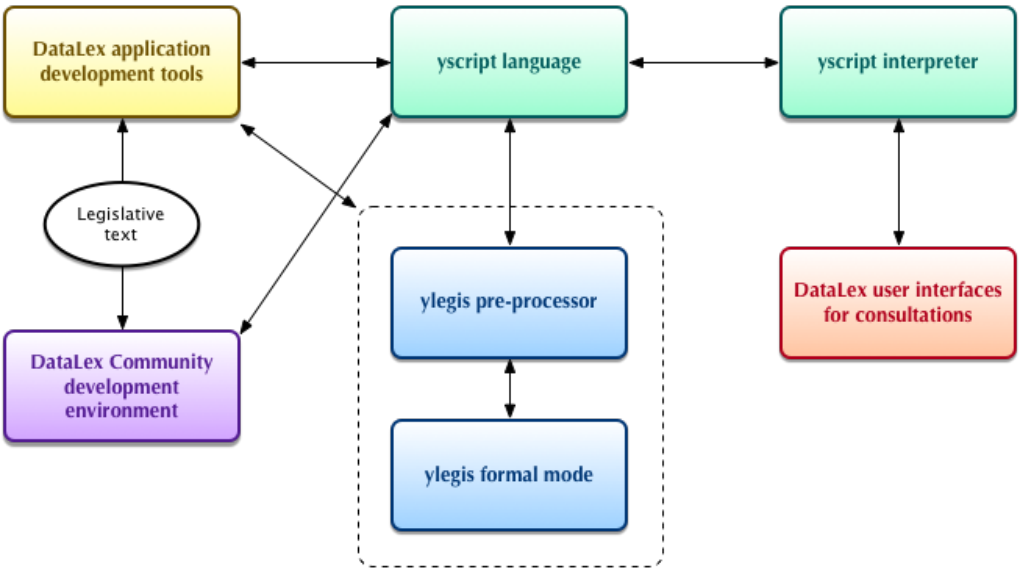
[3] yscript is an extension of the language used in the previous ysh and wysh systems used by DataLex.
[4] GNU Affero General Public License <https://www.gnu.org/licenses/agpl-3.0.en.html>
[5] ylegis is a program which is undergoing constant development, in order to recognise and convert additional legislative structures into code. In many instances, the pre-processed code will need no change, because it is considered to accurately reflect the meaning of the legislation. In many other cases, the pre-processed code will provide a useful 'first draft', but further human editing is required. Further development of ylegis results in more legislative provisions being in the first category after conversion.
[6] DataLex in AustLII Communities< http://austlii.community/foswiki/pub/DataLex/WebHome/ylegis-intro.pdf>

"and:" and "or:". This enables code to be written which is very close to the natural language (English) version of the legislation, but it can be run by the yscript interpreter.

5. **The DataLex application development tools** – All development tools to assist users to develop yscript applications are aggregated on one site.[7] Using this tool, users can import statutory text directly from AustLII, then either manually edit the text to produce yscript code, or use the *ylegis pre-processor* to convert it yscript code. The yscript code can be error-checked for cross-references and for fact translations, using tools provided. It can be edited further if necessary. A consultation can then be run using the default user interface to test by inspection how well the code works.

6. **The default user interface for consultations** – The DataLex default user interface for consultations resulting from use of the yscript interpreter on code developed in yscript has these features. Questions, Facts, Conclusions, and Reports are all generated from the code-base and user-provided facts, in understandable form (in English), and are available on screen at all times. Facts can be deleted ('Forget?'), and questions then re-asked; Conclusions can be explained ('How?); and reasons for Questions requested ('Why?), generated in the same manner. The system also uses all information available to it, from the code-base and user-supplied facts, to suggest other relevant Related Materials (from AustLII's databases). At the end of the consultation, a composite explanation of the final result, and of all the steps necessary for it to be reached, is displayed and may be exported to word processing or other programs for use. Alternative user interfaces have also been developed, requiring no alteration to the yscript code. Examples are in 'Utilising AI in the Legal Assistance Sector'.

7. **DataLex Community development environment** – Located within AustLII Communities, the DataLex Community pages provides a wiki-like editing environment, which inserts automated links between legislation or cases cited in yscript code to the cited legislation or cases on AustLII (or other LIIs). It also inserts the links from consultations being run using the default user interface.



[7] DataLex Application Development Tools page <www.datalex.org/dev/tools>

Conceptual elements of the DataLex approach

The 'DataLex approach' to the desirable approach to developing rules as code is to some extent built-in to various elements of the DataLex toolkit explained above. In many respects, however, it is not technically determined, but is instead a set of recommendations for how to use these tools. The key conceptual elements involved in building rules as code apps with yscript are:

- *yscript* uses a quasi-natural-language 'English-like' syntax, which is easy to learn and use, and produces natural English-like dialogs (consultations) and explanations without any textual 'baggage' in addition to its rules. It generates questions and explanations 'on the fly'. It is well-suited for representing legislation and other rules which are comprised of a structured set of propositions.

- yscript supports different modes of coding, but for most 'law as code' applications, declarative coding (with both backward-chaining and forward-chaining rules) is preferred.

- yscript is best used to represent what rules **say** rather than what they **do.**

- yscript allows (but does not require) isomorphic (one to one) representation of legislation and other types of rules. Isomorphic representation is important for transparency, explanation and maintainability.

- The combination of quasi-natural-language representation, and adherence to isomorphism, means that *yscript* code can be directly created and maintained by lawyers.

- Code-bases of legislation cannot resolve the correct interpretations of the open-textured terms in the legislation. It is necessary for them to provide access to the source documents where those terms are used, to enable users to research what they mean. The DataLex Community environment automates such linkages when yscript code is run.

Open access to the DataLex toolkit

Because AustLII is based in the free access to law movement, and because it considers that it is necessary for the software and code-bases that it develops for AI and law to be able to contribute to the public domain, there are various aspects of its approach that involve open access.

- The software involved should be available for analysis, and for use by others. The yscript interpreter is available under an Affero GPL licence.

- Code-bases should be available for inspection by anyone, so that their fidelity to the legislation and its interpretation can be checked by anyone who wishes to do so. The code for numerous examples of DataLex applications that have been developed as yet is available publicly,[8] and covers such topics as Australia's foreign relations Act, the News Media and Digital Platforms Mandatory Bargaining Code, the Community Gaming Regulation 2020 (NSW), the Modern Slavery Act 2018 (Cth), and disqualification from standing for federal Parliament.

[8] 'AustLII's DataLex AI platform and its relevance to Law Faculties – The DataLex Development Environment' *ALAA Conference*, 4-6 July 2021<https://ssrn.com/abstract=3878729>.

Conclusions – an increasingly valuable approach

The DataLex approach is a rapidly developing set of tools and concepts with which to build rules as code applications. With the addition of the ylegis preprocessor, it is conducive to building large scale applications involving substantial legislative texts, or collections of texts. It can also be extended to other types of rules such as codes of practice, business rules and standards. Integration with AustLII's databases provides a means of dealing with open texture issues. Various interfaces to the same code-base make a variety of real-world applications and use cases possible. While further research and development are needed for all aspects of the DataLex approach, they demonstrate its potential and value.

Further references

Mowbray, Chung & Greenleaf 'Utilising AI in the Legal Assistance Sector – Testing a Role for Legal Information Institutes' LegalAIIA '19, June 17, 2019, Montréal, Québec, Canada, <https://ssrn.com/abstract=3379441>

Greenleaf, Mowbray & Chung 'Building Sustainable Free Legal Advisory Systems: Experiences from the History of AI & Law' (2018) 34(1) *Computer Law & Security Review*, < https://ssrn.com/abstract=3021452>

Mowbray *Coding in yscript* (AustLII, May 2021) <http://Austlii.community/foswiki/pub/DataLex/WebHome/ys-manual.pdf >

Mowbray, Greenleaf & Chung *AustLII's DataLex Developer's Manual* (AustLII, June 2021) <http://austlii.community/foswiki/pub/DataLex/WebHome/datalex-developers-manual.pdf>