JOURNAL OF
CROSS-DISCIPLINARY RESEARCH
IN COMPUTATIONAL LAW [CRCL]

The Uncanny Valley of Computable Contracts: Assessing Controlled Natural Languages for Computable Contracts

Florian Idelberger[*]

Abstract

Automated legal relationships of one form or another are the future. One does not have to listen to legal tech enthusiasts to come to this conclusion. In some limited settings, such automated legal relations are already here. These embedded, automated legal relations can be as seemingly innocuous as a social media app that implicitly encodes its privacy policy within its computer code. One avenue to make these relations more accessible is the combination of controlled natural languages (CNLs, languages based on existing languages such as English but more restrictive) and programming languages, creating a CNL that reads like (for example) English but can be used to create executable computable contracts. These work by reducing the complexity of the natural language to a manageable size by only allowing a limited set of syntax and semantics. While CNLs are more precise and more suitable for such a task than machine-learning-based natural language processing, they also have limitations, especially in their inherently limited complexity and when learning and writing them. This contribution sketches the historical development of controlled natural languages and their relation to programming languages and then assesses how valuable CNLs are for computable contracts. In this process, I describe the specific property of CNLs, namely, that they are often easy to read but hard to master, referring to the 'Uncanny Valley of Computable Contracts' by way of analogy to the hypothesised phenomenon described for human reactions to humanoid androids.

Keywords: computable contracts, controlled natural language, contract automation, human machine interface, AI and Law

Replier: Emma Tosch • Northeastern University, e.tosch@northeastern.edu.

Journal of Cross-disciplinary Research in Computational Law
© 2025 Florian Idelberger
DOI: pending
Licensed under a Creative Commons BY-NC 4.0 license
www.journalcrcl.org

[*] Postdoctoral Researcher, Center for Applied Legal Research, Karlsruhe Institute of Technology and Researcher at FIZ-Karlsruhe (Leibniz Institute for Information Infrastructure), florian.idelberger@kit.edu.

Introduction

In this paper, I present evidence for the overarching problems that CNLs (Controlled Natural Languages) targeting legal applications face when trying to combine human and machine accessibility. These problems make CNLs hard to learn and write without extensive help, which inventors often fail to provide or only develop as an afterthought. This phenomenon leads to CNLs that read like natural language but are hard to learn, write and master; the more so, the harder they are to distinguish from a natural language without being equal. As a result, compared with general programming languages, data representation methods, and graphical 'no-code' interfaces, the usage and acceptance of such CNLs are very low. I hypothesise this low usage and acceptance is due to their unfortunate in-between and uncanny nature, between natural language and programming language. I named this phenomenon the 'uncanny valley' of computable contracts using CNLs, after the phenomenon described by Japanese roboticist Masahiro Mori and later adopted in popular culture.[1]

Contract Automation, in one form or another, has existed for decades, at its most straightforward in the form of standard form contracts.[2] Thoughts about legal automation have been around even longer.[3] Programming languages and restricted or outright invented languages for human reception have also existed longer than legal automation. So far, contract automation has frequently focused on document automation or data analysis.[4] Invented languages often focus on internal logic or perceived improvements over existing natural languages, seldom on accessibility, and seldom on the legal domain.[5] Programming languages are currently at an impasse, where non-natural languages are reasonably user-friendly but very unfamiliar to new users. Most innovations focus on improving the compiler and enforcing secure memory use (as an example) but not on making programming languages more accessible.[6] In contract automation, there has been some recent work on using CNLs to model a language for this limited domain. Examples include Logical English,[7] and Lexon,[8] and similar attempts have been made by CodeX at Stanford.[9] Other comparisons of formalisms for the representation of law or contracts have been written by Ma[10] and Clack,[11] for example.

To explain my hypothesis and the name I chose, I selected existing CNLs targeting contracts and law to provide examples for comparison that show evidence for the hypothesis. This process is a qualitative evaluation of those languages through experimentation and usage, allowing me to provide much more detailed examples and empathetic commentary than with a survey or an empirical evaluation without usage.

Methodology

In this paper, I will describe the relationship between CNLs and computable contracts, and the hypothesis of the 'uncanny valley of computable contracts', sketched above, in more detail based on historical developments and literature.

To show the differences and similarities and to elicit an intuitive understanding in the reader of what these CNLs are capable of (and what they are not capable of), I use observational methodology. This method is well established in sociology and used there when looking at 'law

[1] Masahiro Mori, Karl F MacDorman, and Norri Kageki, 'The Uncanny Valley' (2012) 19(2) IEEE Robotics & Automation Magazine 98.
[2] Meng Wong, 'Computable contracts: From Academia to industry' (2018) 2 Rechtshandbuch Legal Tech 315.
[3] Louis O Kelso, 'Does the Law Need a Technological Revolution' (1945) 18 Rocky Mountain Law Review 378.
[4] Wong (n 2) 212.
[5] Arika Okrent, *In the land of invented languages: Adventures in linguistic creativity, madness, and genius* (Spiegel & Grau Trade Paperbacks 2010).
[6] Examples of innovation in that domain are for example the Rust and Go programming languages.
[7] Robert Kowalski and Akber Datoo, 'Logical English meets legal English for swaps and derivatives' (2022) 30(2) Artificial Intelligence and Law 163.
[8] Henning Diedrich, *Lexon: Digital Contracts* (Wildfire Publishing 2019).
[9] Michael Gnesereth, 'Computable Contracts Project' (2015) ⟨http://compk.stanford.edu/⟩.
[10] Megan Ma, 'Writing in Sign: Code as the Next Contract Language?' (2020) Release 1.0 MIT Computational Law Report; Megan Ma and others, 'Deconstructing Legal Text: Object-Oriented Design in Legal Adjudication' (2020) Release 1.0 MIT Computational Law Report.
[11] Christopher D Clack, 'Languages for smart and computable contracts' (2021) ⟨https://arxiv.org/pdf/2104.03764⟩.

in practice' as opposed to the law in an abstract way.[12] In a different way, it was also often employed by Bruno Latour when describing people and their work, machines, or other contexts that he wrote about as 'the observer', keenly aware of all intermediate steps and assumptions, even cautioning against logic as a reliable foundation.[13] Although CNLs are far removed from legal practice, the methodology is still helpful as CNLs (considered as a technology) are comparable in their practical application, as in Kuhn's survey.[14]

The only difference in the present paper is that an observer observes technological systems (artificial languages) through usage instead of participants. Because of these differences, I describe it as an interdisciplinary socio-technical (and legal) approach, as it uses a sociological method on technological systems that are applied to law. That said, this method is only second best to each reader trying the languages out by themselves, which, after some time, would evoke an intuitive understanding (or misunderstanding) much more prominently. For the purpose of this paper, I assume it is sufficient for me to have applied the CNLs and then to tell the reader about it. This tale is sufficient to help get knowledge about these languages from their respective niches into broader research communities.

The observation is backed by manual usage of each of these systems. This observation involved creating an example software evaluation agreement in Attempto Controlled English (ACE)[15] and Lexon[16] for my thesis, creating test agreements for Logical English, and referring to literature and discussion for L4 and Catala.[17] Together, the hypothesis of the 'Uncanny Valley' and the examples of computable contracts using CNLs can give a new perspective and new opportunities for computable contracts as an example of computable law. For the present paper, the observations have been reduced from those made in the thesis.[18] There, I created a software evaluation license in several formalisms and compared them by using and observing them. In this previous study, formalisms were assessed based on the PENS criteria developed by Kuhn for his survey on CNLs[19] and a qualitative assessment of the capabilities of various formalisms for representing computable contracts, based on comparing the capabilities for representing contractual content, the potential for automation, the representation of the contractual process and subsequent business impact, as well as the aesthetics of contracts. To achieve a more reproducible framework for assessment, in future work, the criteria for assessment should be made more stringent and formalised, similar to the PENS system, where each formalism is assessed in precision, expressiveness, naturalness, and simplicity on a scale from 1 to 5. Due to time and scope constraints however, this is not possible within this paper.

This system was not used here, but a follow-up study could also observe the usage of these technologies by practitioners. This evaluation could take the form of interviews or hands-on sessions with different groups of people. Groups could, for example, consist of lawyers, judges, legal academics, and laypersons. Such a study would, however, require an enormous amount of preparation and resources to be valuable and of sufficient size. Thus, the current observation of CNLs in use is inherently more subjective but informed by the PENS scale and the qualitative evaluation of the additional evaluation factors.

Structure

In order to have a common frame of reference, the concept of computable contracts, as used in this paper, is described

[12] Marc Simon Thomas, 'Teaching Sociolegal Research Methodology: Participant Observation: Special Issue on Active Learning and Teaching in Legal Education' (2019) 14(14) Law & Method.
[13] Graham Harman, 'The importance of Bruno Latour for philosophy' (2007) 13(1) Cultural studies review 31.
[14] Tobias Kuhn, 'A survey and classification of controlled natural languages' (2014) 40(1) Computational linguistics 121.
[15] Florian Idelberger, 'step21/computable-contracts: Turing - Cleanroom Release' (Zenodo July 2022) ⟨https://doi.org/10.5281/zenodo.6877324⟩.
[16] Because it was uncertain when or if Lexon would be made available publicly and because it was thus uncertain whether reviewers and readers could check the Lexon based system and results, I did not include the Lexon version in the thesis.
[17] See the discussion of L4 and Catala below.
[18] Florian Idelberger, 'The Uncanny Valley of Computable Contracts: Analysis of Computable Contract Formalisms with a Focus towards Controlled Natural Languages' (PhD thesis, European University Institute 2022) (not yet public, available upon request from the author).
[19] Kuhn, 'A survey and classification of controlled natural languages' (n 14).

after this introduction. Then, the historical background of invented and controlled languages is traced via glimpses into the past, which shows that much of what is present and problematic in current CNLs is hardly new. Thereafter, I present some general remarks on the syntax, semantics, and logic of CNLs; whereafter, I present one general CNL (Attempto Controlled English), two that specifically target contracts (Logical English and Lexon, both at least initially mainly targeted contracts) and one that focuses more on law with L4. For comparison with non-CNLs targeting law, Catala is described as a programming language specifically for law. In the end, the concept of the 'Uncanny Valley' of CNLs for computable contracts is presented and is analysed and extrapolated in more detail. Specifically, I focus on the usage of CNLs as an interface and the meaning of CNLs as legalism.

The present paper traces separate current and historical developments of invented and controlled languages which were intended to reduce ambiguity and improve accessibility for humans and machines. This process concludes that all CNLs for law have problems in terms of accessibility, intuitive logic/understanding, and therefore learning, and often also with expressivity due to their restrictive rules. These flaws confine their potential usage to very limited circumstances or as a stop-gap measure right now.

Contracts

A contract is an agreement between one or more parties, for example, regarding a specific sale, which then results in an obligation to perform delivery and make payment. Unless otherwise specified by law, such as is often the case for real estate transactions, contracts can be in any form. Contracts are generally used to record business transactions and personal agreements.[20] They do not only have a legal and a business function, however, but also a social function, independent of enforcement and thus independent of computability. This social function exists whether considering computable contracts or non-computable ones. For natural language contracts, socio-legal scholars such as Macneil characterised this social function and functioning as relational contracts, where content matters less, and the relationship between the parties is the most important.[21] In other terms, Allen described the contractual stack, where, in the case of computable contracts, separate layers above the code or contract text might specify intent, offering default interpretations that might be relied on in dispute scenarios.[22] A graphical interface could be an additional layer in this model. In many cases, human oversight or, following Macneil,[23] the relational aspect of contracts is still necessary.

Computable Contracts

Various terms are used in the literature on 'computable' contracts depending on the author and their background. Throughout this paper, I will use the term 'computable contract', first established by Harry Surden in 2012. The purpose of this paper is not to discuss terminology, so it suffices to say that I find this term is the least fraught with other connotations, such as ideology or a specific technology (such as smart contracts), while at the same time, it is not too vague (such as e-contract or algorithmic contract). Surden describes his computable contracts as 'data-based' contracts, and for any computable contract, this is a necessary foundation, as data is irreplaceable for any proper computable contract.[24]

The usage of computable contracts is most often advocated for due to cost savings on transaction costs through

[20] Giesela Rühl, 'Smart (Legal) Contracts, or: Which (Contract) Law for Smart Contracts?' in Benedetta Cappiello and Gherardo Carullo (eds), *Blockchain, Law and Governance* (Springer International Publishing 2021).

[21] Ian R Macneil, 'Reflections on relational contract' [1985] (H. 4) Zeitschrift Für Die Gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics 541.

[22] Jason G Allen, 'Wrapped and stacked:'smart contracts' and the interaction of natural and formal language' (2018) 14(4) European Review of Contract Law 307.

[23] Macneil (n 21).

[24] Harry Surden, 'Computable Contracts' (2012) 46 University of California Davis Law Review 629, 639.

streamlined processes and getting rid of mundane tasks such as form filling.[25] Other factors in advocating for their usage are the improved and facilitated interaction between human and machine actors, especially when using a CNL.[26] This facilitation of interaction comes about through facilitating access to users and machines while making explicit legal or business rules that would otherwise only be implicit in the application code or in hard-to-access electronic or paper documents. Surden further describes the role of computable contracts as an ex-ante prima-facie assessment, in opposition to an ex-post review of a legal contract when problems occur.[27]

Generally, computable contracts can be applied where sufficient data is available for their performance and if they are sufficiently discrete (as opposed to relational). Existing work by Surden and the guidance document, 'Developing a Legal Specification Protocol,' for computable contracts from CodeX both focus on specific domains as primary targets for computable contracts, specifically financial contracts and insurance.[28]

Historical Background On Languages

Historically, inventive minds worldwide felt the urge to invent their own language.[29] This process either involved inventing everything, including sounds and characters, or at least partially basing the new language on existing languages, such as sourcing word parts from one or many existing languages.[30]

The inventors of these languages had various reasons and methods for their endeavours. Up to the 19th and into the 20th century, the belief that a better language could lead to world peace, enlighten feeble human minds, help connect with God or improve language learning was widespread.[31] The last-mentioned was the most realistic and is still relied on as a motivation today in Basic English and other subsets for other languages, such as Leichte Sprache for simpler German and Français Fondamental for easier French.[32]

Other invented languages that still survive today are those that found a particular community that practices and enjoys them. The most well-known of these is Esperanto, which according to Okrent's research, is the most prolific and the only invented language that managed to attract a certain community size and a corresponding culture in that language.[33]

Today's invented languages in the form of CNLs have much more modest and practical objectives, but nonetheless, similarities can be observed. Mainly, both past and present invented or controlled languages struggled with making their inventions accessible to new learners and had difficulty bridging the gap of keeping their purpose (such as a strict logic or no ambiguity) while also getting interested parties to use the language, and at the same time writing down language grammar concisely and intuitively.[34] These languages that came before modern CNLs are not direct predecessors to modern CNLs, but they still serve as an illustration that not all problems are new or confined to them.

[25] Surden (n 24) 688–689, 696; Wong (n 2) 213, 221.
[26] Surden (n 24) 694; Inari Listenmaa and others, 'An NLG pipeline for a legal expert system: a work in progress' (2021) ⟨https://arxiv.org/abs/2107.02421⟩.
[27] Surden (n 24) 678, 679.
[28] Oliver R Goodenough, 'Developing a Legal Specification Protocol: Technological Considerations and Requirements' (2019) ⟨https://law.stanford.edu/publications/developing-a-legal-specification-protocol-technological-considerations-and-requirements/⟩; Surden (n 24).
[29] Goodenough (n 28); Surden (n 24).
[30] Okrent (n 5) 15, 110–115.
[31] ibid 114, 122.
[32] Gudrun Kellermann, 'Leichte und Einfache Sprache–Versuch einer Definition' (2014) 64(9-11) Aus Politik und Zeitgeschichte 7; Dominique Klinger and Georges Daniel Véronique, 'La grammaire du Français fondamental: Interrogations historiques et didactiques' [2006] (36) Documents pour l'histoire du français langue étrangère ou seconde.
[33] Okrent (n 5) 51, 57, 62.
[34] ibid 68, 96, 100, 125.

Current Controlled Natural Languages

Currently, there are several CNLs targeting legal contracts or law, which are at the same time also active in the sense that they are usable, reasonably present in literature, and under active development. For the present paper, I selected Attempto Controlled English (ACE),[35] Logical English,[36] Lexon,[37] L4[38] and Catala.[39] ACE here serves as an example of a very heavily developed and researched CNL, even though it does not directly target contracts or the law. Logical English and Lexon were selected as CNLs that directly target contracts and law. Catala, on the other hand, was selected because it serves as an example of a programming language used in the domain of law that does not try to emulate natural language via a CNL.

Syntax and Semantics of Controlled Natural Languages

In languages of any kind, syntax describes the structure and the rules of the language, and semantics describe its meaning. These terms and general concepts are used universally, whether when discussing natural languages, programming languages, or other formal or artificial languages such as CNLs.

In his survey of CNLs, Kuhn gives a longer definition of CNLs, stating that they are based on one natural language, have restrictions in vocabulary, syntax, and semantics but retain some natural language properties for ease of understanding. In addition, he says whereas natural languages in active use evolve organically, CNLs are explicitly constructed and invented.[40] Based on these properties and further observations, he developed the PENS scale to compare CNLs.[41] This scale measures precision, expressivity, naturalness, and simplicity. Each criterion uses easily evaluated descriptions; for example, simplicity (or complexity) is measured in the length of pages it takes for a full description of the language without the vocabulary. For example, in comparison with a natural language, he assigns English the score $P^1E^5N^5S^1$, which features a low precision (P^1) but high expressivity and high naturalness (N^5). In comparison, propositional logic has high precision but low expressivity with $P^5E^1N^1S^5$. Propositional logic also features the lowest naturalness and high simplicity, whereas English has low simplicity, *i.e.*, high complexity.[42] The CNLs discussed here fall somewhere in between, likely similar to COBOL (a programming language) with $P^5E^2N^2S^3$, though with higher naturalness.

On a related note, Wyner et al. compiled a report on CNLs that listed necessary and possible considerations for designing and evaluating CNLs, where they distinguish between generic properties, high-level properties, design properties, and linguistic properties.[43] On a generic level, they focus on the intended user base, intended purposes, and domain specificity. The ease of reading, writing, ease of learning, and whether the CNL is predictable and unambiguous are also stressed as important factors.[44]

Logic

All CNLs here have the following properties in common when being translated to programming languages – they have a simple logic model that is mostly reliant on Prolog, as in the case of Logical English, or on the underlying pro-

[35] Norbert E Fuchs, Uta Schwertel, and Rolf Schwitter, *Attempto Controlled English (ACE) - Language Manual 99.03* (techspace rep, Institut für Informatik der Universität Zürich 1999).
[36] Robert Kowalski, Jacinto Dávila, and M Calejo, 'Logical English for legal applications' in *XAIF, Virtual Workshop on Explainable AI in Finance* (2021).
[37] Lexon Foundation, 'Lexon-Rust' (2020) ⟨https://gitlab.com/lexon-foundation/lexon-rust⟩.
[38] Listenmaa and others (n 26) 1, 4.
[39] Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko, 'Catala: a programming language for the law' (2021) 5(ICFP) Proceedings of the ACM on Programming Languages 1.
[40] Kuhn, 'A survey and classification of controlled natural languages' (n 14) 123.
[41] ibid 125–132.
[42] ibid 139.
[43] Adam Wyner and others, 'On controlled natural languages: Properties and prospects' in Norbert E Fuchs (ed), *International workshop on controlled natural language* (2009) 3.
[44] ibid 4.

gramming language, in the case of Lexon. Both use variants of first-order logic. This is sufficient for many programming languages but not for obligations and permissions such as in deontic logic.

Many functions of computable contracts which conform to Surden’s conception and are capable of being implemented as a Deterministic Finite Automaton (DFA) as described by Goodenough, are sufficiently represented in code by if-then logic.[45] If-then logic is a key feature of programming languages. In logic theory it is an aspect of first-order logic. First-order logic is often also called predicate logic and can be described as an extension of propositional logic. Where propositional logic consists of statements and their relations, first-order logic adds variables, quantifiers, and more complex relations. First-order logic distinguishes itself from higher-order logic, which for example, can also have quantifiers over functions and predicates.

CNLs targeting Contracts

Attempto Controlled English (ACE) ACE is a general-purpose CNL developed from 1995 onwards at the University of Zürich by Norbert E. Fuchs and colleagues. From there, development diversified into specific application domains and tooling.[46]

Examples of this are uses for knowledge representation, formal reasoning, rules systems with AceRules specifically targeting rule-setting and APE (Attempto Parsing Engine).[47] APE is the main program that makes ACE accessible by checking its correctness on loading a file and optionally transforming it to other ontological representations or paraphrasing the text. In my experiments, I mainly used the paraphrasing functionality to test if the entered ACE text matched what it was meant to express after transformation and ‘understanding’ by the program.

ACE is a subset of English but has a limited vocabulary (though it is user-extendable), and only very specific grammar constructs are permitted when writing ACE. Sentences consist of a subject, verb, complements, and optionally adjuncts. These are terms from linguistics, where a complement is a part completing a sentence, and an adjunct is an optional part of a sentence. This part can either be an object, a modifier, or a similar addition.

Like many formal CNLs, ACE only supports simple present, and it always needs an article or a quantification, even if natural English would not. In addition, ‘this’, ‘these’ and any word containing ‘any’ is prohibited.[48] Furthermore, according to its manual, ACE only supports active voice[49] and expressions in the third person singular. It does not support imperatives, plural noun phrases modal logic or deontic logic verbs such as ‘must’, ‘shall’ and ‘may’.[50] Some of these restrictions, such as plural noun phrases, are necessary for disambiguation. If the plural noun phrase ‘noun phrases’ were used, it would be difficult (or maybe impossible) to have a rule on what this referred to. The phrase could refer to a collection of phrases, at least one, or to specific instances of noun phrases mentioned previously. This explanation shows that there are good reasons why these restrictions are in place – but at the same time, they also highlight the deceptive similarity to natural language.[51]

In terms of sentences supported, ACE supports not only simple sentences but also composite sentences, such as the special constructs of ‘if’ and ‘of’ sentences. Of these, for contracts or programming constructs, ‘if-sentences’ are the most important due to their propensity for rules.[52]

[45] Surden (n 24) 647, 665; Mark D Flood and Oliver R Goodenough, ‘Contract as automaton: Representing a simple financial agreement in computational form’ (2022) 30(3) Artificial Intelligence and Law 391.

[46] Fuchs, Schwertel, and Schwitter (n 35).

[47] https://github.com/Attempto/APE.

[48] Fuchs, Schwertel, and Schwitter (n 35) 26; APE authors, ‘APE - Illegal Words’ (2021) ⟨https://github.com/Attempto/APE/blob/61c9b264dd/prolog/lexicon/illegalwords.pl⟩.

[49] Fuchs, Schwertel, and Schwitter (n 35) 2, 3.

[50] A noun phrase is a phrase that has a noun at its head and has the same function as a noun.

[51] Adam Wyner, ‘From the Language of Legislation to Executable Logic Programs’ in Michał Araszkiewicz and Krzysztof Płeszka (eds), *Logic in the Theory and Practice of Lawmaking* (Springer International Publishing 2015) 420–423.

[52] Fuchs, Schwertel, and Schwitter (n 35) 25.

Lastly, ACE also supports query sentences; examples are yes/no questions and questions that start with the query words 'who' and 'what'.[53]

The ACE manual is much more extensive, but the rules presented here are meant to give an overview from a user perspective.[54] While I assume it is possible to condense the manual, these examples of the most basic rules give a glimpse of what is necessary for a language such as ACE to avoid ambiguity while still reading like natural English.

While there are attempts to formalise the ACE grammar in new ways, such as by providing an abstract grammar, these help with studying and research but not with increasing accessibility and general usage.[55] Hoefler, for example, describes an abstract grammar based on the software implementation of ACE, the Attempto Parsing Engine (APE), and uses a grammar syntax closely mirroring Definite Clause Grammars (DCGs) in the logic programming language Prolog.[56]

For CNLs, domain knowledge is one feature that can make them more accessible so that some of the supported grammar and vocabulary are derived and potentially can be intuited by a user based on what they know about the domain in question.[57] Due to its generality, ACE does not feature domain grounding. The specific usage of ACE could provide such grounding or adapt in other ways, as in the case of AceRules, which adapted ACE for rules as code (though it does not use this term) and allowed usage of different semantic models depending on the envisaged application.[58]

Hoefler says, 'It has become evident that a controlled natural language like ACE can combine the advantages of natural and formal languages ... [and] contribute to ... a successful interaction and communication between domain specialists and software engineers.'[59] So far, Hoefler's positive findings have not come to fruition, as evidenced by lack of usage in addition to my observations on usage.

Figure 1 below contains a contract clause from an ACE example derived for my thesis, which compared implementations of a software evaluation license in different formalisms.[60] ACE is the most natural of all currently usable CNLs as it is most closely modelled after natural language. Still, many often-used phrases and constructs, such as the past tense and passive voice, are not supported. These restrictions do not have to be a hindrance, as the results can work or be adapted to work in many cases. Still, in an overall text, it can lead to an unnatural flow of text.[61] In ACE, the notation 'n:' can be used to designate additional words directly inside a text (in this case, a noun), whereas otherwise, vocabulary is defined in separate files.

In my experiments, I then used APE to process the ACE text and derive a paraphrased version of the text. On careful reading, this shows whether the text is accurately represented in ACE because if it is not, the paraphrased version will contain errors that distort the intended meaning in many cases. This process shows most errors but still requires a careful reading of the results. From the example below, it can be deduced that paraphrasing also results in less natural text, which harkens back to ACE's Prolog roots regarding the logic and argumentation used.

Because ACE aims for high naturalness, it has considerable complexity. This makes it a fascinating language and ACE could be an excellent way to design accessible and usable computable contracts. However, it was not designed to be

[53] Fuchs, Schwertel, and Schwitter (n 35) 34.
[54] Version 3.0 of the language manual has 81 pages.
[55] Stefan Hoefler, *The Syntax of Attempto Controlled English: An Abstract Grammar for ACE 4.0* (techspace rep, Institut für Informatik der Universität Zürich 2004).
[56] ibid 2.
[57] Wyner and others (n 43) 3.
[58] Tobias Kuhn, 'Acerules: Executing rules in controlled natural language' in Massimo Marchiori, Jeff Z Pan, and Christian de Sainte Marie (eds), *International Conference on Web Reasoning and Rule Systems* (2007).
[59] Hoefler (n 55) 18.
[60] Idelberger, 'The Uncanny Valley of Computable Contracts: Analysis of Computable Contract Formalisms with a Focus towards Controlled Natural Languages' (n 18).
[61] Kuhn, 'A survey and classification of controlled natural languages' (n 14) 138.

```
/* Article 3 */

The Licensee and a n:sublicensee may not publish at least
one comment that is not approved by the Licensor.
The approval of at least one comment needs to be given before
the publication. If the Licensee publishes at least one comment
without the approval of the Licensor then the Licensee must remove
the comment within 24 h.
```

Figure 1: *An example clause in ACE depicting a licensing clause*

```
It is false that it is admissible that they publish at least
1 comment X1 and that it is false that Licensor approves
the comment X1. If Licensee publishes at least 1 comment X1
without an approval of Licensor then it is necessary that
Licensee removes the comment X1 within 24 h.
```

Figure 2: *The paraphrased ACE example*

executable or data-based, as all output formats focus on knowledge representation or reasoning about facts, similar to Prolog.[62] In his work, Wyner came to a similar conclusion, describing the limitations of ACE as its strength but also seeing its limitations for complex domains such as the law.[63] With enough modifications, ACE could be modified to be an executable language, but this would probably mean additional restrictions and rules would need to be added. The specific modifications necessary depend on the application domain.[64]

Logical English Logical English is the brainchild of logician Robert Kowalski, who has been influential in applying logical reasoning concepts to specific domains, especially logic programming and the law[65] Logical English is based on the logic-programming language Prolog, which Kowalski helped grow. The roots of Kowalski’s work on the combination of logic programming and natural language go back to the 1990s.[66] In the past, he used Prolog to model the British Nationality Act and worked on the basics of logic programming.[67]

The peculiarity of Logical English is that it is described as ‘syntactic sugar’ on top of Prolog.[68] This thin layer has the effect that it is very limited in the grammatical structures that it supports, even more so than ACE. In this context, ‘support’ means that it can differentiate or even understand them. In Logical English, it is possible to use many kinds of constructs if templates define them, but they will only make it easier for humans to recognise the constructs, though not in the way computers understand them. At the same time, Logical English will feel familiar to Prolog programmers, as the layer between the CNL and the Prolog programming language is relatively thin.

Logical English at first targeted contracts (and still uses the domain and GitHub organisation LogicalContracts), but by now, it has a broader focus that includes law and regulation.[69]

[62] APE authors (n 48).
[63] Wyner and others (n 43) 424.
[64] ibid.
[65] Kowalski, Dávila, and Calejo (n 36); Kowalski and Datoo (n 7).
[66] Robert Kowalski, ‘English as a logic programming language’ (1990) 8(2) New Generation Computing 91.
[67] Marek J Sergot and others, ‘The British Nationality Act as a logic program’ (1986) 29(5) Communications of the ACM 370; Robert Kowalski, ‘Predicate logic as programming language’ in *IFIP congress* (1974) vol 74.
[68] Kowalski, Dávila, and Calejo (n 36) 1.
[69] LogicalContracts authors, ‘Logical English - Knowledge Base - Citizenship’ (2022) ⟨https://github.com/LogicalContracts/LogicalEnglish/blob/2bc845870afa3b8fc57ac1e5fd0cd7ffab13a106/kb/0%5C_citizenship.pl⟩.

```
en("the target language is: prolog.
  the templates are:
  the terms of *a contract* are met,
  ...
  the knowledge base minicontract includes:
  ...
  A contract C is a valid contract if
  the contract C is signed by the service provider known as X.
  scenario one is:
  ... The payment schedule is of type staggered payment schedule.
  query one is:
  which contract is a valid contract.
```

Figure 3: *An example document structure for Logical English*

Figure 3 below shows the current structure of a Logical English document and an abbreviated example. First, the templates section demonstrates that the structure of each sentence must be defined as a template for the recognition of any sentence.[70] Asterisks denote defined variables. Then the templates defined in the first section are instantiated in the 'knowledge base' section, where they constitute the actual rules. The next two important sections are scenarios and queries, both of which can contain multiple different items. A scenario is a specific set of circumstances defined in addition to the knowledge base. As an example, when someone is appointed to a position or a given variable is either true or false, this can be set in a scenario. Queries are questions that combine a question word, a variable, and the sentence parts defined in templates and the knowledge base. Queries and scenarios can then be combined to 'answer' questions, similar to how this would work in the original Prolog.

This querying functionality is impressive. When using a carefully constructed knowledge base in conjunction with an appropriately phrased scenario and query, it answers questions in a natural style and explains how it reached its answer. This functionality is especially useful when combined with a special 'answer' predicate that tries to explain an answer in CNL based on the text templates it finds when looking for answers. For this feat, when looking for answers, it repeats the parts (Prolog predicates) in reverse order and turns them back into sentences.

The use of templates for each sentence exemplifies that Logical English is 'syntactic sugar'. Compared to other languages, it has only a few logic-based constructs, phrases, and some math that are predefined and mapped to the underlying Prolog. Everything else depends on user-defined templates. In my view, for legal applications, this is a strength and a weakness at the same time. It is a strength because it allows for many natural-sounding sentences without being restricted by pre-existing vocabulary or constructs. On the other hand, it requires all those sentences to be predefined and might give a false sense of equivalency to natural language if it is not apparent to a user that the system distils the elaborate sentence they devised down to a long, but still simple Prolog term of the form $a(b^1, \dots, b^N)$.

This distillation to the essentials can be seen in Figure 4 below. It depicts a Prolog term with two arguments from the knowledge base above.

Overall, representing a contract in Logical English allows users to capture many expressions for computable contracts. If Logical English is paired with an additional system for execution, it can also create computable rules similar to smart contracts. It also is much more accessible than

[70] Kowalski, Dávila, and Calejo (n 36) 1.

[71] Thomas F Gordon, 'Some problems with prolog as a knowledge representation language for legal expert systems' (1987) 3(1) International Review of Law, Computers & Technology 52, (as one example).

```
    the_terms_of(A_CONTRACT, are_met).
```

Figure 4: *The rule from the Figure 3 above, translated into a Prolog compound term*

Prolog, which was used a lot in the past for explorations of computable law.[71]

Lexon Lexon is a CNL that targets contracts and was born out of the renewed interest in contract automation that followed the blockchain and smart contract hype from 2017 onward. Consequently, its first target language was Solidity, a lower-level[72] language used to write smart contracts for certain blockchain systems.[73] It is a domain-specific language as it directly targets contracts and the law. When targeting Solidity, it is grounded in certain primitives supplied by the Solidity language and its underlying environment, which other languages, such as Logical English, do not have.

The grammar of Lexon, while not overly complicated, is hard to describe concisely. The basic document structure follows the structure of a contract document.

In the example of Figure 5, the general structure of a Lexon document is visible. The document is suited to the terminology and look of a natural language contract by using a readable header, a natural language preamble, and terms such as 'Terms', 'Contracts', and 'Clause(s)'. At the same time, however, these structures are mapped onto the underlying language in a specific way. This mapping is less of a problem if the language is just used for prototyping, such as in the case of Metafor,[74] but since the languages under analysis are meant to be computable directly, this matters as users need to be aware of the limitations and particularities of a particular output language.

In its original application, where Lexon targets Solidity, the whole contract document maps onto a Solidity contract instance, which in other programming languages might be akin to a class in object-oriented programming. A class in programming languages is a template to easily spawn instantiations with common functions and base parameters.

Then, the '`TERMS`' section maps onto the constructor of the smart contract. Each clause maps onto a function of the contract. '`CONTRACTS per`' allows for an instantiation of a contract per user, for example. Also, before '`TERMS`', it is possible to define and assign entities outside of the constructor, as in any Solidity program. For example, an expression of '`X is Y`' then assigns a type, and '`X fixes Z`' (or other verbs are also possible) then sets who may set a value for a particular variable.

This format does make smart contracts written in Lexon very easily readable at first glance. However, so far, it does not work in reverse. As a result, it is currently not possible to translate Solidity smart contracts to Lexon, which could be very useful. When Lexon is applied to smart contracts and thus when translating to Solidity, the use of legal or non-programming terms hides the fact that the CNL is merely an interface for the underlying smart contract language. This fact poses three major problems. One is that, in principle, it still requires at least a non-negligible familiarity with Solidity as a smart contract programming language. Second, even if that familiarity exists, the CNL does not allow for the fine-grained control that is necessary in many cases, especially when dealing with high value smart contracts and high complexity. Third, there is no quick and easy way to determine if the output of a certain translation does what it says in English or whether this code is safe. To some degree, the latter two points can be mitigated by hard-coding certain common cases and prohibiting other constructs.

[72] Referring to a blockchain system (Ethereum Virtual Machine) in this case, which uses its own bytecode.
[73] Solidity Developers, 'Ethereum/Solidity' (2015) ⟨https://github.com/ethereum/solidity⟩.
[74] Hugo Liu and Henry Lieberman, 'Metafor: Visualizing stories as code' in *Proceedings of the 10th international conference on Intelligent user interfaces* (2005). This tool does not produce fully functional code, but code from natural language input, which can for example be used for brainstorming.
[75] https://github.com/smucclaw/dsl.

```
LEX: Contract Name.
LEXON: 0.3.x
AUTHOR: Lex Lawyerperson
PREAMBLE: Explanation goes here.
TERMS:
"Licensor" is a person.
"Licensing Fee" is [an amount].
The Licensor fixes the Licensing Fee.
...
CONTRACTS per Some Name:
...
"Some Entity" is a person.
...
CLAUSE: Payment
The Licensee pays the Licensing Fee to the Licensor,
and pays the Breach Fee into escrow.
The License is therefore Paid.
```

Figure 5: *An abbreviated example of a Lexon document*

CNLs targeting Law and Regulation

L4 L4 is a language targeting law and regulation and is currently under active development at Singapore Management University by serial legal entrepreneur Meng Wong.[75] It is a domain-specific language (DSL) but features an optional CNL for natural language generation.[76]

This CNL part is built on top of Haskell and Grammatical Framework (GF), a rule-based language framework to map and translate between languages and its Resource Grammar Library. GF is a software framework and toolset written in Haskell, a functional, statically typed language.[77] A big difference to Prolog, as used for ACE and Logical English, is the use of types, which are a core feature of GF.[78] This approach results in a system much more oriented towards specific types and categories that build on top of each other. In my view, this can make a language more accessible, which is also supported by the importance of domain dependence, as identified by Wyner et al.[79]

It is, in principle, possible to retroactively build types into a Prolog language, but in the cases under examination, this was not done. In addition, L4 is much more than merely 'syntactic sugar'. This sets some ground rules that make certain rules and use cases easier, as the possibility to have types of words is already built in through the programming language types.

In addition to being built on top of GF, L4 uses GF's Resource Grammar Library (RGL), which allows it to harness the basic syntax and the foundations of word and sentence construction from over 30 languages.[80] Due to the capabilities of GF in conjunction with the RGL, of all CNLs, L4 is the CNL that tries to capture the complexities of natural languages most completely and is the most ambitious in terms of grounding and scope.

[76] Inari Listenmaa and others, 'Towards CNL-Based Verbalization of Computational Contracts' in Tobias Kuhn and others (eds), *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)* (2023).
[77] Aarne Ranta, *Grammatical framework: Programming with multilingual grammars* (vol 173, CSLI Publications, Center for the Study of Language and Information Stanford 2011); Aarne Ranta and others, 'Abstract syntax as interlingua: Scaling up the grammatical framework from controlled languages to robust pipelines' (2020) 46(2) Computational Linguistics 425.
[78] Listenmaa and others (n 76) 2.
[79] Wyner and others (n 43) 3.
[80] Grammatical Framework, 'GF Resource Grammar Library (RGL)' (2008) ⟨https://github.com/GrammaticalFramework/gf-rgl⟩.

```
# optional CNL descriptions
lexicon UnauthorizedSharingFees
@ "involves sharing fees with unauthorized persons" MayAcceptAppointment
@ "{LegalPractitioner} may accept an appointment in {Business}"
```

Figure 6: *An example showing the complementary CNL used by L4, where it describes two different programming descriptors in more detail. The example is from a paper by Listenmaa and colleagues (n 76), and used for illustrative purposes.*

Figure 6 shows snippets of the CNL as attached to DSL descriptors, describing the unauthorised sharing of fees and whether a legal practitioner may accept an appointment. These parts are not used as a core language but as an additional descriptor to describe or generate natural language descriptions for rules written in the DSL itself. This connection between DSL and CNL is a different take on literate programming and making contract code accessible.

Programming Languages targeting Law and Regulation

The selection of the languages presented in this article is partially based on the development activity, relevance and novelty of those languages. Taking account of these factors, only Catala is listed here, but historically, there were many more programming languages targeting law and regulation. In the rules-as-code movement, Python or other general programming languages are also used for similar purposes.[81]

Catala Catala is a recent development that foregoes CNLs or natural language likeness for a programming language, a domain-specific language that targets law and regulation.[82] A key feature of Catala is built-in support for literate programming.[83] This support means that it supports the inclusion of the natural language version of a legal clause above or below the computable clause, in a form of literate programming. Jupyter Notebooks use the same technique for data science.[84]

In terms of logic, it is described as using default logic as opposed to predicate logic (also often known as first-order logic). Predicate logic is, for example, used by Prolog, which APE and Logical English use under the hood.

Default logic is similar to defeasible logic, which has often been proposed in the Law and AI community as being well suited to legal applications due to its ability to resolve conflicting rules and having exceptions and defaults.[85] Default logic has default rules in case no conditions apply. According to Antoniou, who compared both, defeasible logic is directly deductive, and default logic is based on alternative worldviews.[86] While these both use different approaches, the result is similar, and default logic, as used in Catala, can be seen as an extension of earlier work in law and AI on defeasible logic for legal applications.[87]

The example in Figure 7 shows a section of Catala from its introductory paper.[88] Based on this paper, Catala explicitly targets laws and regulations, and this is exemplified by providing facilities to model the complexities and interdependence of existing laws, such as the scope shown in the example above and a wholly declarative nature allow-

[81] Brenda Wallace, 'When software and law are the same thing' (2019) ⟨https://2019.pycon-au.org/talks/when-software-and-law-are-the-same-thing⟩; Jason Morris, Rules as Code: How Technology May Change the Language in Which Legislation Is Written, and What It Might Mean for Lawyers of Tomorrow, 'ABATECHSHOW' (2021); OpenFisca Aotearoa, 'BetterRules' (2022) ⟨https://github.com/BetterRules/openfisca-aotearoa⟩.
[82] Merigoux, Chataing, and Protzenko (n 39); The Catala developers, 'The Catala compiler and tooling' (2020) ⟨https://github.com/CatalaLang/catala⟩.
[83] Merigoux, Chataing, and Protzenko (n 39) 9.
[84] Mary Beth Kery and others, 'The story in the notebook: Exploratory data science using a literate programming tool' in *Proceedings of the 2018 CHI conference on human factors in computing systems* (2018).
[85] Grigoris Antoniou and David Billington, 'Relating defeasible and default logic' in *Australian Joint Conference on Artificial Intelligence* (2001).
[86] ibid.
[87] Grigoris Antoniou and others, 'Embedding defeasible logic into logic programming' (2006) 6(6) Theory and Practice of Logic Programming 703.
[88] Merigoux, Chataing, and Protzenko (n 39).
[89] ibid 10.

```
scope Section121SinglePerson:
  rule requirements_ownership_met under condition
    aggregate_periods_from_last_five_years of personal.property_ownership >= 730 day
```

Figure 7: *A partial example describing US income tax legislation*

ing for out-of-order computation as the law may require due to interdependences.[89]

The Uncanny Valley

Generally, CNLs are easy to read and sometimes not distinguishable from a natural language (English in this case) but are hard to write and master. During my research, I found that this phenomenon was already known when writing CNLs and is mentioned by Kowalski in the context of Logical English and by Kuhn more generally.[90] However, with these authors, it is not described from a legal perspective and was not given any particular name.

I propose to call this phenomenon the 'Uncanny Valley of Computable Contracts' when applied to computable contracts. I use the term 'Uncanny Valley' as it reminds me of the hypothesis of the same name used to describe the human response to humanoid robots and virtual characters. Specifically, Masahiro Mori, a Japanese roboticist, hypothesised that – at a certain level of human-likeness – people's willingness to accept humanoid robots would drop sharply (due to the robots appearing strange or 'uncanny') and only increase again when they become almost indistinguishable from humans. I hypothesise that a similar effect applies to learning and writing human-readable and writeable computable contracts using CNLs, and by extension, also to users' acceptance of such CNLs. The proliferation of the term 'Uncanny Valley' of computable contracts should give more visibility in research and literature to this limitation, to the possibilities of computable contracts featuring CNLs, and make the advantages and limitations of CNLs as well as the Uncanny Valley hypothesis as applied to computable contracts accessible to a broader audience.

This hypothesised relationship is approximated in the graph above. Acceptance and ease of use in writing a computable contract increase with a rise in the likeness to natural (legal) language but fall deep into a 'valley' where this increase fails, only increasing again when almost like a natural language. Deepest in the valley are general CNLs, with domain-specific CNLs on the valley's cliffs. Despite domain-specific languages being 'in the valley', they are also the best bet to get out of it. They are 'the best bet' because it is more feasible to write an easily readable and writeable CNL that is intuitively grounded if it is limited to a specific domain. Formalisms and CNLs are placed further to the right on the slope up to the valley as they get more and more 'natural' as evidenced by the possibility to use natural words and sentences and higher up as the hypothesised acceptance by laymen increases. Inside the valley, I differentiate between general and domain-specific CNLs, but they are not placed anywhere specifically, as the differences are too minute in terms of this hypothesis. The shape is also only approximated. More to the right are formalisms that support more elements of natural language.

Similar to this graph, the original hypothesis concerning the response to humanoid androids was merely a hypothesis (albeit with reasoned argumentation). Later on, the 'Uncanny Valley' hypothesis proposed by Mori was empirically tested and confirmed by Mathur and others.[91] Their observed valley, however, differed from the hypothesised one, depending on what was measured and displayed in the graph.

[90] Tobias Kuhn, 'A principled approach to grammars for controlled natural languages and predictive editors' (2013) 22(1) Journal of Logic, Language and Information 33, 33–34; Kowalski, Dávila, and Calejo (n 36) 5.
[91] Maya B Mathur and others, 'Uncanny but not confusing: Multisite study of perceptual category confusion in the Uncanny Valley' (2020) 103 Computers in Human Behavior 21, 26–27.

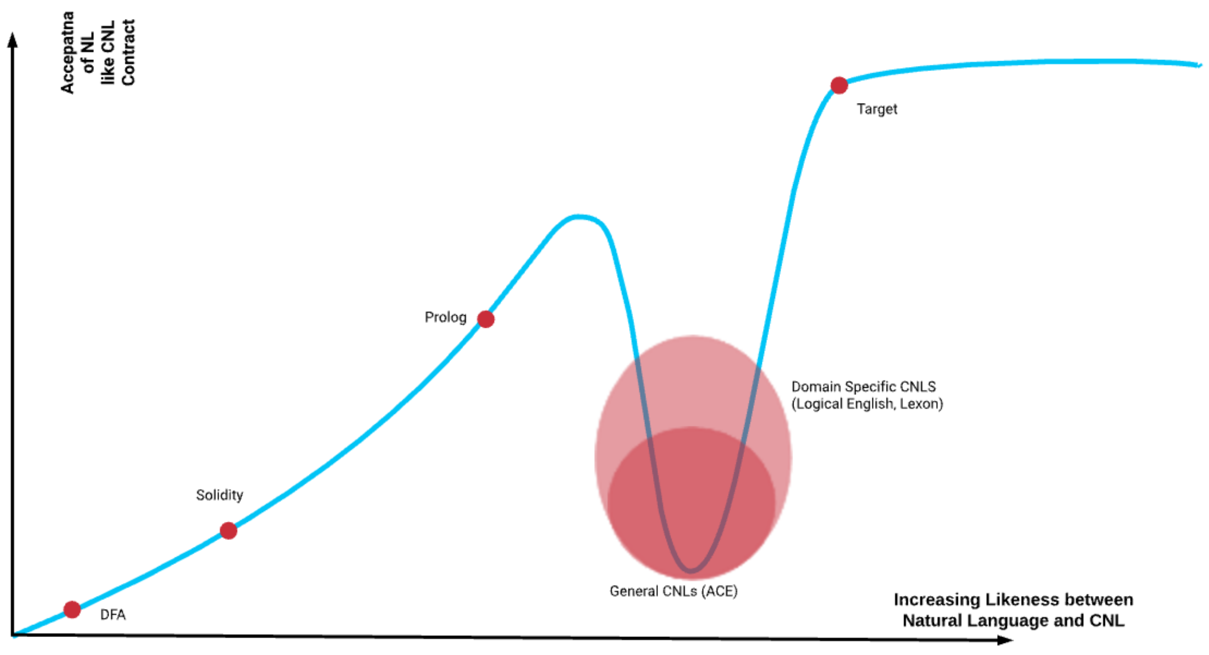

Figure 8: *The 'Uncanny Valley' of CNLs for Computable Contracts – An ascending graph describing the growth of acceptance of CNLs for contracts as they become more like natural language. The graph shows a deep valley before the plateau of being like a natural language.*

Analysis

I posit that while looking like natural (legal) language is a net positive for the adoption and comprehension of computable contracts, there is an 'Uncanny Valley', a point on the transition from programming language to indistinguishable from natural legal language where a computable contract language looks like a natural language contract, like any ordinary contract, but where it is harder to comprehend and write. This 'valley' then provokes a dislike and unacceptance in users and stakeholders, not due to looks, but due to difficulty of use. Instead of helping adoption and understanding of the computable contract, this CNL then makes it harder to understand and even creates confusion. Below, I will further contextualise my findings and the hypothesis of the 'Uncanny Valley' by describing and categorizing CNLs as an interface and as 'legalism' or in other words an attempt to enforce legal certainty through code.

CNLs as an Interface

Primarily, text, whether in a legal contract or a computable contract, and whether a natural language text or a computable text, is an interface. The limitations and the subsequent confusion described above occur for CNLs within the textual interface because the compatibility and comprehension by man and machine alike of a CNL are achieved by limiting grammatical structures, words, and logical constructs that can be employed when writing the computable contract, that is, when using the CNL as an input device.[92] While this usage is not a given, it is the main one highlighted by projects such as Lexon and Logical English. These observed limitations and confusion mean that many rules need to be memorised and applied for competent use when writing new content in a CNL. These rules bear no relation to the existing language knowledge of the user, are often not derivable according to certain fundamental principles and thus have to be memorised (or checked automatically against a written grammar or a compiler). This issue means that using a CNL is like a

[92] Wyner (n 51) 422–424.

completely new language, even though it does not look like it. The rules as to which constructs are allowed or not often do not follow any general, intuitive practice. As a result the writer may be tempted to write in the original natural language, even when the CNL does not permit such usage.[93] Such CNLs, in effect, have to be learned like a foreign language, eliminating a key benefit of using CNLs based on a natural language. This effect is similar to 'false friends' when learning English and makes learning harder. Further aspects, such as technological wariness or incompatibility with contracting rituals, can also contribute to the uncanniness of CNLs for (computable) contracts.[94]

Furthermore, in examples such as Inform7,[95] a natural language-like programming language for programming text adventures, the language is not ambiguous and can be used to program. However, while it is easily readable, it is also very verbose. This verbosity can be circumvented by employing specific keywords and limiting the domain. However, the more general an application is attempted, the more problematic this becomes. As a result, it is much harder to arrive at a general CNL for the description and automation of computable contracts. Beyond the technical and linguistic challenges, it also has to gain acceptance and be easy to learn.[96] This relation between generality and domain limitation for ease of use is so far unsolved and might be inherent.

Last, I will discuss the use of a CNL as an output device. The example of Inform7 can also help us learn about CNLs as an output interface, as it shows a similar problem for output to a CNL – any output based on programming code will likely be very verbose once it is necessarily expanded to form natural-like sentences. Moreover, whether it accurately conveys the meaning of the computer code would have to be studied separately. The closest comparator is the L4 DSL and the L4 project's plans for isomorphic natural language generation from L4 code.[97] In any case, the resulting language would not necessarily lead to a proper natural language document, as it would have to be isomorphic to the computer code.[98]

The above sums up the confusion and issues surrounding the use of CNLs for contracts and is also why CNLs sound better in theory or when only reading them as opposed to when researchers or users try to use them.

Some shortcomings of CNLs could be improved by following Wyner et al.'s high-level and design properties, though only to a limited degree. Some limitation is inherently necessary for a CNL, since there are conflicts between different design objectives such as the CNL being both unambiguous for a computer, and having a high degree of expressivity.[99]

In summary, the use of CNLs as an interface for computable contracts present various issues, such as the plethora of rules that need to be memorised for competent use, rules often bearing no relation to existing language knowledge or following intuitive patterns. Additionally, domain limitation can make CNLs easier to learn, but it also makes them less useful. Lastly, in many instances of proposing usage of a CNL for computable contracts there is no guarantee that the meaning of the contract is accurately conveyed and overall, CNLs often sound better in theory than in practice and when using them actively.

CNLs as Legalism

Codifying the law, whether by programming language or CNL, is generally done to make law computable, as then it can be embedded it into a business process, a process of legal adjudication, or contractual enforcement.[100] The use of computable contracts is a symptom of codifying more and more agreements, of trying to make contractual processes predictable and rigid. The same thus applies to

[93] Wyner and others (n 43) 3–5.
[94] Joanne P Braithwaite, 'Standard form contracts as transnational law: evidence from the derivatives markets' (2012) 75(5) The modern law review 779.
[95] Graham Nelson, 'About Inform7' (2006) ⟨http://inform7.com/about/⟩.
[96] In a similar pattern, many inventors in the 19th century sought to create an artificial language from scratch, that was supposed be more logical and just better than natural language, but almost none of them took into account how easy it was to learn.
[97] Listenmaa and others (n 76) 1.
[98] Clack (n 11) 19.
[99] Wyner and others (n 43).
[100] Surden (n 24) 659, 663, 688.

CNLs for computable contracts, and this enforces certain properties on otherwise flexible legal instruments. This reinforcement of legalistic properties can be very welcome, especially when business analysts and economists hear 'reduction of transaction costs'. However, this focus can also lead to a fixation on legal certainty and only looking at short-term costs. This tale is especially prevalent in the narratives that helped the blockchain industry grow. In these narratives, blockchain was depicted as offering a counterpart to the squishy and malleable system of law in the real world. An unstoppable codified smart contract would fix right and wrong in contract disputes once and for all.[101]

In a recent article, Laurence Diver described this much more eloquently and termed it 'computational legalism'.[102] He goes on to say that delay, (on which both legality and contracts are built), must be taken into account when digitising law. Otherwise, there is a risk of constraining law to the limits of the code, as Diver says. Others have described similar concerns not with regard to delay but equity or other interpretative methods.[103]

As a result, even when not looking specifically at computation like Diver but at the language used to create computable contracts, concerns about the implications of legalism should be taken into account by all involved. Language developers, especially, too often disregard the restricting implications of legalism or leave them to users to take care of. This realisation does not mean that CNLs for computable contracts are not useful; at the very least, they can make existing code more accessible where there is inherent business logic, or they can allow a legal contract to be created and governed more explicitly, as opposed to only implicitly through code.

The 'Uncanny Valley', as a gap between the perceived and actual textual representation of contracts, also is related to this concept of legalism, as a CNL can be a technical measure to solidify a legalistic approach to interpretation and necessarily restricts expressivity to make the language non-ambiguous and executable. While CNLs specifically, and computable contracts generally, can also improve accessibility and can in theory support delay and other affordances in law, legalism, as Diver calls it, is a strong driver for many CNL inventors.[104] This driving force motivated and motivates both historical language inventors[105] and the developers of CNLs targeting law, regulation, or contracts, as both groups favoured and favour a rigid rule-based logic. The lure of reduced transaction costs only adds to the allure of legalism.[106]

In qualifying CNLs as a form of legalism, I base my argument on several points. Mainly, this stems from a vision of law which affords legal certainty through codification, a vision hailed by CNL inventors as delivering affordances that are in contrast with the affordances of non-computable law. Additionally, CNLs can make the relationship between code and law more explicit, and therefore more accessible. Lastly, the 'Uncanny Valley' as a gap between perceived and actual textual representation of computable contracts also signifies the legalistic aspects of CNLs.

Conclusion

In the paper, based on observation and experimentation with CNLs for computable contracts, I analysed the socio-legal and techno-legal aspects of these CNLs in a qualitative way, based on how they read and write and by reference to existing work and literature. I argued that CNLs

[101] Primavera de Filippi, 'Blockchain Technology as an Instrument for Global Governance' (SciencesPo - Chaire Digital, Gouvernance et Souveraineté 2020 2020); Primavera De Filippi, Chris Wray, and Giovanni Sileno, 'Smart Contracts' (10 Internet Policy Review 2021) ⟨https://policyreview.info/glossary/smart-contracts⟩; Marcella Atzori, 'Blockchain Technology and Decentralized Governance: Is the State Still Necessary?' (2017) 6 Journal of Governance and Regulation.

[102] Laurence Diver, 'Computational legalism and the affordance of delay in law' (2021) 1(1) Journal of Cross-disciplinary Research in Computational Law.

[103] Maren K Woebbeking, 'The impact of smart contracts on traditional concepts of contract law' (2019) 10 J. Intell. Prop. Info. Tech. & Elec. Com. L. 105.

[104] Henning Diedrich, *Lexon Bible: Hitchhiker's Guide to Digital Contracts* (Wildfire Publishing 2020) 120.

[105] Okrent (n 5) 69.

[106] Surden (n 24) 688.

can make computable contracts more readable. However, they are not very helpful for making computable contracts or business logic more accessible or user-friendly and often they do not seem to be designed with actual users in mind. This seems similar to how early language inventors failed at making their languages usable. For this concept of hard-to-use CNLs, I coined the phrase ‘The Uncanny Valley of Computable Contracts’.[107] This term depicts a valley in the acceptance of more natural-language-like languages, which lie in an unfortunate middle ground between a programming language or a simple CNL and a natural language. This middle ground makes them great to look at but challenging to use and master. Naming this concept in such a way gives an easier way to reference the phenomenon in subsequent literature and may connect formerly disconnected strands of literature. Using a CNL merely for the output of a computable contract is also feasible and would work much better with fewer headaches, but so far, no projects have proposed this as a main usage scenario. A further scenario is a comprehensive contractual stack that can accommodate not only code and natural language text but also graphical user interfaces.

References

Allen JG, ‘Wrapped and stacked:‘smart contracts’ and the interaction of natural and formal language’ (2018) 14(4) European Review of Contract Law 307.

Antoniou G and Billington D, ‘Relating defeasible and default logic’ in *Australian Joint Conference on Artificial Intelligence* (2001).

Antoniou G and others, ‘Embedding defeasible logic into logic programming’ (2006) 6(6) Theory and Practice of Logic Programming 703.

Aotearoa O, ‘BetterRules’ (2022) ⟨https://github.com/BetterRules/openfisca-aotearoa⟩.

APE authors, ‘APE - Illegal Words’ (2021) ⟨https://github.com/Attempto/APE/blob/61c9b264dd/prolog/lexicon/illegalwords.pl⟩.

Atzori M, ‘Blockchain Technology and Decentralized Governance: Is the State Still Necessary?’ (2017) 6 Journal of Governance and Regulation.

Braithwaite JP, ‘Standard form contracts as transnational law: evidence from the derivatives markets’ (2012) 75(5) The modern law review 779.

Clack CD, ‘Languages for smart and computable contracts’ (2021) ⟨https://arxiv.org/pdf/2104.03764⟩.

developers TC, ‘The Catala compiler and tooling’ (2020) ⟨https://github.com/CatalaLang/catala⟩.

Diedrich H, *Lexon: Digital Contracts* (Wildfire Publishing 2019).

— *Lexon Bible: Hitchhiker’s Guide to Digital Contracts* (Wildfire Publishing 2020).

Diver L, ‘Computational legalism and the affordance of delay in law’ (2021) 1(1) Journal of Cross-disciplinary Research in Computational Law.

Filippi P de, ‘Blockchain Technology as an Instrument for Global Governance’ (SciencesPo - Chaire Digital, Gouvernance et Souveraineté 2020 2020).

Filippi PD, Wray C, and Sileno G, ‘Smart Contracts’ (10 Internet Policy Review 2021) ⟨https://policyreview.info/glossary/smart-contracts⟩.

Flood MD and Goodenough OR, ‘Contract as automaton: Representing a simple financial agreement in computational form’ (2022) 30(3) Artificial Intelligence and Law 391.

Foundation L, ‘Lexon-Rust’ (2020) ⟨https://gitlab.com/lexon-foundation/lexon-rust⟩.

Framework G, ‘GF Resource Grammar Library (RGL)’ (2008) ⟨https://github.com/GrammaticalFramework/gf-rgl⟩.

Fuchs NE, Schwertel U, and Schwitter R, *Attempto Controlled English (ACE) - Language Manual 99.03* (techspace rep, Institut für Informatik der Universität Zürich 1999).

Genesereth M, ‘Computable Contracts Project’ (2015) ⟨http://compk.stanford.edu/⟩.

Goodenough OR, ‘Developing a Legal Specification Protocol: Technological Considerations and Requirements’ (2019) ⟨https://law.stanford.edu/publications/developing-a-legal-specification-protocol-technological-considerations-and-requirements/⟩.

[107] More precise would be ‘The Uncanny Valley of Controlled Natural Languages for Computable Contracts’.

Gordon TF, 'Some problems with prolog as a knowledge representation language for legal expert systems' (1987) 3(1) International Review of Law, Computers & Technology 52.

Harman G, 'The importance of Bruno Latour for philosophy' (2007) 13(1) Cultural studies review 31.

Hoefler S, *The Syntax of Attempto Controlled English: An Abstract Grammar for ACE 4.0* (techspace rep, Institut für Informatik der Universität Zürich 2004).

Idelberger F, 'step21/computable-contracts: Turing - Cleanroom Release' (Zenodo July 2022) ⟨https://doi.org/10.5281/zenodo.6877324⟩.

— 'The Uncanny Valley of Computable Contracts: Analysis of Computable Contract Formalisms with a Focus towards Controlled Natural Languages' (PhD thesis, European University Institute 2022).

Kellermann G, 'Leichte und Einfache Sprache–Versuch einer Definition' (2014) 64(9-11) Aus Politik und Zeitgeschichte 7.

Kelso LO, 'Does the Law Need a Technological Revolution' (1945) 18 Rocky Mountain Law Review 378.

Kery MB and others, 'The story in the notebook: Exploratory data science using a literate programming tool' in *Proceedings of the 2018 CHI conference on human factors in computing systems* (2018).

Klinger D and Véronique GD, 'La grammaire du Français fondamental: Interrogations historiques et didactiques' [2006] (36) Documents pour l'histoire du français langue étrangère ou seconde.

Kowalski R, 'Predicate logic as programming language' in *IFIP congress* (1974) vol 74.

— 'English as a logic programming language' (1990) 8(2) New Generation Computing 91.

Kowalski R and Datoo A, 'Logical English meets legal English for swaps and derivatives' (2022) 30(2) Artificial Intelligence and Law 163.

Kowalski R, Dávila J, and Calejo M, 'Logical English for legal applications' in *XAIF, Virtual Workshop on Explainable AI in Finance* (2021).

Kuhn T, 'Acerules: Executing rules in controlled natural language' in M Marchiori, JZ Pan, and C de Sainte Marie (eds), *International Conference on Web Reasoning and Rule Systems* (2007).

— 'A principled approach to grammars for controlled natural languages and predictive editors' (2013) 22(1) Journal of Logic, Language and Information 33.

— 'A survey and classification of controlled natural languages' (2014) 40(1) Computational linguistics 121.

Listenmaa I and others, 'An NLG pipeline for a legal expert system: a work in progress' (2021) ⟨https://arxiv.org/abs/2107.02421⟩.

Listenmaa I and others, 'Towards CNL-Based Verbalization of Computational Contracts' in T Kuhn and others (eds), *Proceedings of the Seventh International Workshop on Controlled Natural Language (CNL 2020/21)* (2023).

Liu H and Lieberman H, 'Metafor: Visualizing stories as code' in *Proceedings of the 10th international conference on Intelligent user interfaces* (2005).

LogicalContracts authors, 'Logical English - Knowledge Base - Citizenship' (2022) ⟨https://github.com/LogicalContracts/LogicalEnglish/blob/2bc845870afa3b8fc57ac1e5fd0cd7ffab13a106/kb/0%5C_citizenship.pl⟩.

Ma M, 'Writing in Sign: Code as the Next Contract Language?' (2020) Release 1.0 MIT Computational Law Report.

Ma M and others, 'Deconstructing Legal Text: Object-Oriented Design in Legal Adjudication' (2020) Release 1.0 MIT Computational Law Report.

Macneil IR, 'Reflections on relational contract' [1985] (H. 4) Zeitschrift Für Die Gesamte Staatswissenschaft/Journal of Institutional and Theoretical Economics 541.

Mathur MB and others, 'Uncanny but not confusing: Multisite study of perceptual category confusion in the Uncanny Valley' (2020) 103 Computers in Human Behavior 21.

Merigoux D, Chataing N, and Protzenko J, 'Catala: a programming language for the law' (2021) 5(ICFP) Proceedings of the ACM on Programming Languages 1.

Mori M, MacDorman KF, and Kageki N, 'The Uncanny Valley' (2012) 19(2) IEEE Robotics & Automation Magazine 98.

Morris J, Rules as Code: How Technology May Change the Language in Which Legislation Is Written, and What It Might Mean for Lawyers of Tomorrow, 'ABATECHSHOW' (2021).

Nelson G, 'About Inform7' (2006) ⟨http://inform7.com/about/⟩.

Okrent A, *In the land of invented languages: Adventures in linguistic creativity, madness, and genius* (Spiegel & Grau Trade Paperbacks 2010).

Ranta A, *Grammatical framework: Programming with multilingual grammars* (vol 173, CSLI Publications, Center for the Study of Language and Information Stanford 2011).

Ranta A and others, ‘Abstract syntax as interlingua: Scaling up the grammatical framework from controlled languages to robust pipelines’ (2020) 46(2) Computational Linguistics 425.

Rühl G, ‘Smart (Legal) Contracts, or: Which (Contract) Law for Smart Contracts?’ in B Cappiello and G Carullo (eds), *Blockchain, Law and Governance* (Springer International Publishing 2021).

Sergot MJ and others, ‘The British Nationality Act as a logic program’ (1986) 29(5) Communications of the ACM 370.

Solidity Developers, ‘Ethereum/Solidity’ (2015) ⟨https://github.com/ethereum/solidity⟩.

Surden H, ‘Computable Contracts’ (2012) 46 University of California Davis Law Review 629.

Thomas MS, ‘Teaching Sociolegal Research Methodology: Participant Observation: Special Issue on Active Learning and Teaching in Legal Education’ (2019) 14(14) Law & Method.

Wallace B, ‘When software and law are the same thing’ (2019) ⟨https://2019.pycon-au.org/talks/when-software-and-law-are-the-same-thing⟩.

Woebbeking MK, ‘The impact of smart contracts on traditional concepts of contract law’ (2019) 10 J. Intell. Prop. Info. Tech. & Elec. Com. L. 105.

Wong M, ‘Computable contracts: From Academia to industry’ (2018) 2 Rechtshandbuch Legal Tech 315.

Wyner A, ‘From the Language of Legislation to Executable Logic Programs’ in M Araszkiewicz and K Płeszka (eds), *Logic in the Theory and Practice of Lawmaking* (Springer International Publishing 2015).

Wyner A and others, ‘On controlled natural languages: Properties and prospects’ in NE Fuchs (ed), *International workshop on controlled natural language* (2009).

A reply: Uncanny to Whom? Usability in light of purpose and provenance of domain-specific languages for law

Emma Tosch • Northeastern University, e.tosch@northeastern.edu

The Uncanny Valley of Computable Contracts surveys the use of five formal languages for writing legal documents, performing an analysis similar to a cognitive walkthrough in the human-computer interaction (HCI) literature,[108] while drawing an explicit connection to the 'uncanny valley' phenomenon of human-robot interaction.[109] The languages under study range from controlled natural languages to programming languages; all are suitable for encoding legal agreements in terms of their *ability* to express legal concepts.

The significance of this work lies in its emphasis on the user experience when attempting to read and encode legal reasoning using these domain-specific languages (DSLs). However, the description of these languages' 'uncanniness' leads us to ask: uncanny for whom? We will address this question in two parts. First we will address why one would design a DSL in the first place. Then we will discuss the role of the designer and their target audience.

Motivations for designing DSLs. First we observe that all of the reasoning DSLs provide could also be directly encoded in a general purpose programming language. There are three common axes along which one might justify the introduction of a DSL: usability, efficiency and correctness. Programs written in DSLs are typically shorter with built-in domain-specific abstractions – often in the form of keywords or operators that correspond to some core concept in the domain – that improve their usability. DSLs are also higher-level than their general purpose language counterparts, often automating lower-level decision making; this automation can optimise generated code, leading to performance improvements (*i.e.*, using fewer resources, such as time, space, energy, etc.). Finally, many DSLs can be proven to have certain desirable properties by virtue of their formal semantics or their use of provably-correct software components. Thus, DSLs may come with guarantees where any program written in the language is correct with respect to some property, *simply by virtue of being written in the language.*

In the context of the law, a key attraction of DSLs is that one small change to the written law ought to correspond to one small change in the DSL encoding. Any change to the DSL encoding then propagates to the executable code without further intervention from a person. This relationship is not necessarily preserved when we encode the law directly in a general purpose programming language. Thus, the value of DSLs lies in their *specificity*.

DSLs should lower the barrier for domain experts to perform specific tasks. Practising law – *i.e.*, professionally interpreting or writing legal texts – is regulated, often requiring years of training and accreditation.[110] Despite the professionalisation of practising law, it is still common for lay people to, for example, write or enter into contracts without the involvement of a lawyer.[111] Thus, while domain experts could be lawyers or other policy-makers, they could also be businesspeople, scientists or other experts whose work requires interpreting or writing legal documents.

Background of DSL designers. Given the variability of who writes and interprets legal documents, the 'uncanniness' of using formal DSLs for law may increase as the actual community of users drifts away from the original intended community of users. It can also be helpful to

[108] Clayton Lewis and Cathleen Wharton, 'Cognitive Walkthroughs' in *Handbook of Human-Computer Interaction* (Elsevier 1997).
[109] Masahiro Mori, Karl F MacDorman, and Norri Kageki, 'The Uncanny Valley' (2012) 19(2) IEEE Robotics & Automation Magazine 98.
[110] Harold L Wilensky, 'The professionalization of everyone?' (1964) 70(2) American journal of sociology 137.
[111] Tess Wilkinson-Ryan and David A Hoffman, 'The common sense of contract formation' (2015) 67 Stanford Law Review 1269.

contextualise some of these languages in terms of who originally designed them and for what purpose. We will now consider two of the languages featured in this survey: Attempto Controlled English (ACE) and Catala.

ACE is not explicitly a legal DSL, which the author of this article acknowledges. ACE was originally developed by researchers from natural language processing and linguistics for use by software and web developers or other professional looking to formally encode knowledge.[112] Critically, ACE was developed in the artificial intelligence (AI) community, rather than in the programming languages (PL) community. This is not unusual; most formal languages in use were not developed by PL researchers, but rather by experts or practitioners for whom a new formal language was the solution to some extant domain problem. It is increasingly the case that when PL researchers design languages, they do so as domain outsiders, requiring input from target users in order realise the full spectrum of benefits that DSLs can provide (*i.e.*, usability, efficiency and correctness).[113]

Catala is explicitly a legal DSL, not a constrained natural language. Although Catala's authors worked with input from legal experts when designing the surface syntax, they state that it is not their intention for lawyers to use Catala 'unaccompanied'.[114] Catala has some elements of 'naturalness', but unlike the CNLs featured, Catala is very obviously a formal language. As such, the author considers it less 'uncanny'. Recast in HCI terms, Catala uses formality as a kind of *design friction* that prevents users from experiencing a false sense of security.

Unique challenges in DSLs for Law. While the author of this article attributes some 'uncanniness' to 'false friends' found in controlled natural languages, the legal domain can itself contribute to the problem. After all, legal texts are already written in dialects which, despite appearances, may not conform to the diction, grammar or semantics of the apparent languages[115]. Thus any analysis of the usability of formal languages for the law needs to identify whether it is the domain (*i.e.*, the law) or the language design causing confusion. I would contend that any DSL that, for example, targets English-speaking lawyers as users is unsuitable for English-speaking non-lawyers on account of the fact that, despite appearances, the latter group lacks fluency in the appropriate language.

Once a language has been designed, implemented and released into the world, it becomes an object of study. This is the context for the author's work. While the author's single-perspective view on the usability of formal languages for law has value, further quantitative and qualitative assessment from stakeholders should be considered before determining whether the languages under study are too 'uncanny' for the target population (*i.e.*, non-lawyers) to use.

References

Chasins SE, Glassman EL, and Sunshine J, 'PL and HCI: better together' (2021) 64(8) Communications of the ACM 98.

Coblenz M and others, 'PLIERS: a process that integrates user-centered methods into programming language design' (2021) 28(4) ACM Transactions on Computer-Human Interaction (TOCHI) 1.

Fuchs NE and Kaljurand K, 'Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces' [2006].

Lewis C and Wharton C, 'Cognitive Walkthroughs' in *Handbook of Human-Computer Interaction* (Elsevier 1997).

Mellinkoff D, *The Language of the Law* (Wipf and Stock Publishers 2004).

[112] Norbert E Fuchs and Kaarel Kaljurand, 'Attempto Controlled English meets the challenges of knowledge representation, reasoning, interoperability and user interfaces' [2006] .

[113] Sarah E Chasins, Elena L Glassman, and Joshua Sunshine, 'PL and HCI: better together' (2021) 64(8) Communications of the ACM 98; Michael Coblenz and others, 'PLIERS: a process that integrates user-centered methods into programming language design' (2021) 28(4) ACM Transactions on Computer-Human Interaction (TOCHI) 1.

[114] Denis Merigoux, Nicolas Chataing, and Jonathan Protzenko, 'Catala: a programming language for the law' (2021) 5(ICFP) Proceedings of the ACM on Programming Languages 1.

[115] David Mellinkoff, *The Language of the Law* (Wipf and Stock Publishers 2004).

Merigoux D, Chataing N, and Protzenko J, ‘Catala: a programming language for the law’ (2021) 5(ICFP) Proceedings of the ACM on Programming Languages 1.

Mori M, MacDorman KF, and Kageki N, ‘The Uncanny Valley’ (2012) 19(2) IEEE Robotics & Automation Magazine 98.

Wilensky HL, ‘The professionalization of everyone?’ (1964) 70(2) American journal of sociology 137.

Wilkinson-Ryan T and Hoffman DA, ‘The common sense of contract formation’ (2015) 67 Stanford Law Review 1269.

Author’s reponse

Florian Idelberger

I am grateful to Emma Tosch for her thoughtful reply to my article on *The Uncanny Valley of Computable Contracts*. Tosch’s response somewhat reframes the topics I was addressing. My focus is on Controlled Natural Languages (CNLs). Tosch on the other hand focuses on Domain Specific Languages (DSLs). There is some overlap between the two but they are not the same; a DSL need not be a CNL and vice versa.[116] Tosch considers the differences in usability of DSLs as between lawyers and non-lawyers, and the peculiarities of legal language that can also contribute to uncanniness. In that vein, in relation to DSLs, Tosch asks: Uncanny for whom?

A comparison of DSLs and CNLs Tosch describes the motivations of DSL designers as usability, efficiency and correctness, with which I largely agree. I would only add that for CNLs, which are my main focus, the main feature that sets them apart is the perceived usability aspect of seeming and reading like natural language. The other factors identified by Tosch, like efficiency and correctness are always higher for DSLs that are not CNLs, as they lack the verbosity of natural language. The uncanniness I identified mainly applies to CNLs.

Uncanniness as distance between languages. In the article I point to a ‘knowledge gap’ between CNLs and the natural language on which they are based. This ‘knowledge gap’ confuses the intuition users have built up for a specific natural language or for their domain and leads to uncanniness. On this hypothesis, domain-specificity might be assumed to lower uncanniness for legal experts (whether lawyers or not) since it can decrease this knowledge gap or make it easier for actors to cross it. This assumption aligns with literature and personal experience.[117] My arguments were meant to convey that this domain-specificity can help legal experts work with a particular CNL. However, I acknowledge that this depends very much on the implementation of the CNL and how well it fits into the workflow and use of language of these legal experts. This is because not all legal language is the same, but might differ based on education, personal preferences or for other reasons. If these or other factors prevent the ‘knowledge gap’ from being bridged, then domain-specificity will not alleviate the uncanniness. For example, in the case of Lexon, a CNL which allows smart contracts to be written in legal terms, familiarity with blockchain code and the use of fixed terms grounded in the blockchain system can help ease the uncanniness. However, in my view this mostly helps users who are familiar with smart contract code or the way it adapts legal terms for smart contract code. It may not help all legal experts let alone all users. Moreover empirical research is needed to decide whether such domain-specificity is close enough to the original domain to help legal experts (or others) learn a legal CNL, or whether its potentially different usage of familiar terms only leads to confusion and frustration.

To conclude, the circumstances where domain-specificity is helpful are likely very narrow, as the unique feature of CNLs is their naturalness, and it is difficult to optimize for that usability while also keeping the right domain-specificity and other language metrics. In any case, more research is needed into what exactly defines the uncanniness of CNLs, where it occurs and how it might be reduced.

[116] ACE, one of the examples I refer to in the article, is a CNL but not a DSL.

[117] Benjamin Hoffmann and others, ‘An empirical evaluation of a novel domain-specific language–modelling vehicle routing problems with Athos’ (2022) 27(7) Empirical Software Engineering 180.