Computable Contracts for Insurance:
Establishing an Insurance-Specific Controlled Natural Language - InsurLE

John Cummins[*], Jacinto Dávila[**], Robert Kowalski[***], David Ovenden[****]

Abstract

For the most part, insurance contracts are monolithic, text-based documents, written in unstructured natural language interspersed with technical terms. This means that the many provisions embedded in the contract, both explicit and implicit, are difficult for both humans and computers to understand and process. Efforts to make insurance contracts computable have typically resulted in codified representations of contracts that are separate from the contract wording. This creates the problem of verifying whether or not the contract and the codified representation of the contract have the same meaning. In this paper we address this problem by proposing the use of a computable, controlled natural language, InsurLE, which exposes the internal logic of a contract, while preserving much of the syntactic form of the original wording.

InsurLE is a domain-specific extension of LE (Logical English), which in turn is 'syntactic sugar' for the purely logical subset of the computer language Prolog and other logic programming languages. InsurLE extends LE by incorporating an insurance-specific ontology, restrictive relative clauses and a representation of rules and exceptions that mirrors their representation in existing insurance contract wordings. However, like LE and unlike many existing contract wordings, InsurLE avoids complex syntactic structures, which can make insurance contracts difficult to understand and process. As a consequence, contracts written in InsurLE can not only be executed by computer, but can also be easier for laypeople to understand.

Keywords: computable contract, controlled natural language, Prolog, logic programming, commercial insurance.

[*] Co-Founder of Axiome Partners Ltd and Senior Honorary Research Fellow at University College London.
[**] CESIMO, Universidad de Los Andes.
[***] Professor Emeritus and Distinguished Research Fellow, Department of Computing, Imperial College London.
[****] Chief Underwriting Officer, AXA Commercial Plc.
The authors thank Axiome Partners Ltd for funding the initial stages of this work. The authors are also grateful to members of staff of AXA Commercial in the UK for their feedback and advice, to Miguel Calejo, Preston Carlton, Veronica Dahl and Michael Genesereth for helpful comments on earlier drafts of this paper, and to the Editors, Miriam Goldby and Franziska Arnold-Dwyer for guiding us through the refinement of the paper.

1. Background and General Introduction

Creating a digital future for the insurance industry has been a strategic imperative for over two decades. However, despite the progress that has been made, and the digital solutions that have been deployed, this future has still not been realised. This is because the fundamental requirements for digitalisation have not been adequately addressed, and until they are, a true digital transformation of the insurance industry will lie beyond our reach.[1]

In the non-life context, an insurance policy is essentially a contract in which the insurer promises to indemnify the policyholder for loss caused by an insured peril (by paying or otherwise compensating for all or some of the costs associated with the loss). The contract is documented in the policy wording, a document that can exceed one hundred pages, together with the schedule, which contains the specific details relating to the policyholder and the risk. At the moment, a small fraction of these details (usually those in the schedule) are fed, and often re-fed, into a number of platforms and systems across the insurance value chain in order to 'digitalise' the process, while the majority of the contract remains operationally inert, and stored in the form of a text file with a body of unstructured natural language.

We believe that, rather than continuing to build an industry around a set of largely unstructured, monolithic, text-based documents, a much better approach is to make insurance contracts '***computable***', by exposing their internal structure and logic, and representing contracts as structured '***digital objects***'.

The benefits of making insurance contracts computable in this way include: automated (or at least semi-automated) claims handling; a more granular portfolio and exposure management;[2] improved management of new and existing products; more efficient contract management (for both new business and renewals); better communication with customers and brokers; and an overall improvement in customer outcomes.[3] [4]

Computable contracting, understood in this way, represents a fundamental shift of thinking in the insurance industry, as well as an opportunity to reengineer the foundations upon which the insurance industry is built.

In contrast with the approach advocated in this paper, much of the work undertaken to date to make an existing contract computable involves creating a separate computer coded representation of the contract, which does not necessarily reflect the implicit structure and logic of the original contract wording. In short, the 'words' and the 'code' are separate objects. A change made to one object requires an equivalent (but

---

[1] J. Cummins, *Automating (Re)insurance through Computable Contracting* (2020) Version 1.1. https://axiomepartners.com/wp-content/uploads/2020/08/Automating-Reinsurance-through-Computable-Contracting.pdf [accessed 14 November 2024].

[2] 'Granular portfolio and exposure management' concerns establishing a detailed understanding of the potential losses that could arise from a corpus (e.g. thousands) of contracts.

[3] J. Cummins and C. D. Clack, 'Transforming commercial contracts through computable contracting' (2022) 6 (1) *Journal of Strategic Contracting and Negotiation*, https://doi.org/10.1177/20555636211072560 [accessed 13 November 2024].

[4] M. Genesereth, 'A Cure for Health Insurance "Sludge" ', Complaw Corner (2021) Codex: The Stanford Center for Legal Informatics, 2021, online at https://law.stanford.edu/2021/03/31/a-cure-for-health-insurance-sludge/ [accessed 14 November 2024].

separately implemented) change to the other. Furthermore, this separation of words and code creates the challenge of verifying that the two objects are semantically equivalent i.e. that they communicate the same message and that, ultimately, they amount to an identical promise of indemnification.[5]

In this paper, we address this equivalence challenge by proposing the use of a single language, **InsurLE**, to serve ***both*** as the wording ***and*** as a computable codification of an insurance contract. InsurLE is a new **controlled natural language (CNL)**, which, like other CNLs,[6] [7] is designed both to be understandable without technical training, and also to be directly executable as computable code. Our new CNL builds upon an existing CNL, **Logical English (LE)**,[8] [9] [10] [11] [12] which in turn is 'syntactic sugar'[13] for a subset of the **logic programming (LP)** language Prolog.[14]

We argue that the novel approach presented in this paper has the potential both to fulfil the promise of computable contracts and to do so in a way which respects and complements existing insurance industry practices. We will support our argument by presenting examples of insurance contract wordings in InsurLE and by demonstrating how those wordings can be used for such purposes as answering queries about a contract, given information about a real or imaginary scenario.

Answering a query in InsurLE is performed by first translating a contract and scenario into a logical representation in Prolog, and then using Prolog or an extended version of Prolog as a logical reasoner to derive answers as logical consequences of the contract and scenario. The most obvious application of such reasoning is for use by a claims handler, to help process insurance claims. However, the reasoner could also

---

[5] M. Genesereth, 'Contract Definition Language, Complaw Corner' Codex: The Stanford Center for Legal Informatics, 2021, online https://law.stanford.edu/2021/04/07/contract-definition-language/ [accessed 14 November 2024].
[6] F. Idelberger, 'The uncanny valley of computable contracts: analysis of computable contract formalisms with a focus towards controlled natural languages' (2022) Cadmus.eui.eu. online at https://doi.org/10.2870/638745 [accessed 14 November 2024].
[7] T. Kuhn, 'A Survey and Classification of Controlled Natural Languages' (2014) 40 (1) Computational Linguistics, 121 https://doi.org/10.1162/coli_a_00168 [accessed 14 November 2024].
[8] R. Kowalski, English as a logic programming language (1990) 8(2) New Generation Computing, 91 https://doi.org/10.1007/bf03037468 [accessed 14 November 2024].
[9] R. Kowalski, 'Logical English' in Proceedings of Logic and Practice of Programming (LPOP) (2020) online at https://www.doc.ic.ac.uk/~rak/papers/LPOP.pdf [accessed 14 November 2024].
[10] R. Kowalski, J. Dávila and M. Calejo, 'Logical English for legal applications' (2021) XAIF, Virtual Workshop on Explainable AI in Finance online at https://www.doc.ic.ac.uk/~rak/papers/LE_for_LA.pdf [accessed 14 November 2024].
[11] R. Kowalski and A. Datoo, 'Logical English meets legal English for swaps and derivatives' (2022) 30 (2) Artificial Intelligence and Law, 163 https://link.springer.com/content/pdf/10.1007/s10506-021-09295-3.pdf [accessed 14 November 2024].
[12] R. Kowalski, J. Dávila, G. Sartor and M. Calejo, 'Logical English for Law and Education' in: Warren, D.S., Dahl, V., Eiter, T., Hermenegildo, M.V., Kowalski, R. and Rossi, F. (eds) *Prolog: The Next 50 Years. Lecture Notes in Computer Science*, vol 13900. (Springer, 2023). https://doi.org/10.1007/978-3-031-35254-6_24 [accessed 14 November 2024].
[13] In computer science, 'syntactic sugar' is a syntax for symbolic expressions, which hides some of the symbolism and makes the expressions easier to read and understand.
[14] D. S. Warren, V. Dahl, T. Eiter, M. Hermenegildo, R. Kowalski and F. Rossi (eds), *Prolog: The Next 50 Years* (Springer, 2023).

be used by a customer to explore the consequences of selecting different contract options, to help in tailoring a contract to a user's needs, although this may have regulatory implications the consideration of which is outside the scope of this article. Of potentially greater significance for readers of this Journal, the reasoner could also help the drafter of a contract to explore the logical consequences of different wordings.[15]

We developed InsurLE in the context of translating the Public and Products Liability (PPL) wording from an AXA Commercial product[16] into LE, and by showing our initial findings to insurance professionals. They reported that the LE wordings were too clumsy and too wordy for their needs, and that the computations were too restrictive. In response to these criticisms, we have extended the syntax of LE, so that contracts written in InsurLE more closely resemble the style of standard insurance contract wordings. We also extended the LE computation method, so that it can provide conditional answers to queries, when the information about a particular case is incomplete. Although these extensions were developed in the course of our work on the AXA PPL wording, they can also be used both for other insurance contracts and for human-understandable computable contracts more generally.

In contrast with most other CNLs, InsurLE aims to provide a minimal syntax, which maximises naturalness and expressive power, but avoids the syntactic complexities and ambiguities of uncontrolled natural language. For example, in Section 3 we will see that InsurLE does not allow certain syntactic structures which have been found to inhibit readers' understanding and recall of legal texts.[17]

In the main part of this paper, we present the syntax (section 4), semantics (section 5) and computation methods (section 6) of InsurLE, using examples of insurance contract wordings, including the wording of the insuring clause and one of its exclusions from the AXA PPL (sections 7 and 8). We will see how InsureLE builds upon the use of LP rules of the logical form *conclusion if conditions*, how it includes expanded LP rules with restrictive relative clauses in the *conclusions* of LP rules, and how it includes rules of the form *conclusion only if conditions* for expressing exceptions to rules. InsurLE also includes an insurance-specific hierarchy of concepts and associated vocabulary.

But first we survey the relationship of InsurLE to other approaches for making insurance contracts computable (section 2), and we show informally how coverage, exclusions, exemptions and general

---

[15] The case for using symbolic logic to help with drafting and interpreting legal documents was eloquently put by Layman Allen in his 1957 paper, L. Allen, 'Symbolic logic: a razor-edged tool for drafting and interpreting legal documents' (1957) 66 Yale Law Journal, 833). The approach to drafting presented in this paper has similar benefits, with the added advantages that it can be employed without training in symbolic logic, and it can be exercised by experimenting with executable code.

[16] See AXA, 'Business Combined' online at https://www.axaconnect.co.uk/commercial-lines/branch-traded/business-combined/ [accessed 18 December 2024].

[17] E. Martínez, F. Mollica and E. Gibson, 'Poor writing, not specialised concepts, drives processing difficulty in legal language' (2022) 224/105070 Cognition https://doi.org/10.1016/j.cognition.2022.105070 [accessed 14 November 2024].

conditions of insurance contracts are expressed in InsurLE (section 3). We conclude with some directions for future work (section 9).

When writing this paper, we have attempted to avoid assuming that the reader has had any previous background in computing or formal logic. Where we may have failed, we hope that the reader will be able to follow the main thread of the paper, even if the reader may not be able to understand every detail.

2. Making Insurance Contracts Computable – Rationale and Approaches

The InsurLE approach to making insurance contracts computable contrasts with, but is potentially compatible with three other approaches, which have been developed and used more generally in the field of computational law.[18] [19] [20] The first approach, **modularisation**, breaks contracts down into smaller objects and semantically tags these objects to assist with some aspects of automation e.g. search and selection of objects, and user-specific configuration. The second approach, **codified representation**,[21] expresses the implicit logic of a contract wording explicitly as 'code' in a computer language (so that the code exists as a separate object in parallel with the contract). The third approach, **large language models (LLMs)**, generates answers to user-posed queries about an unstructured natural language text, using machine learning trained on massive amounts of data.

Contract modularisation involves breaking contract wordings down (or disaggregating them) into a range of customisable and reusable modular components, thereby facilitating the creation (through reaggregation), amendment and management of 'contract objects' with greater precision and efficiency. With these objectives in mind, content management and document assembly technologies have been successfully deployed in the publishing industry and, more relevantly, for the authoring and management of technical documentation for at least two decades. It is only more recently, however, that these technologies have started to be deployed in the insurance industry, although current deployments tend to be of a proprietary nature and are not based upon open standards. Arguably, further advancement in the modularisation of contracts in commercial insurance will be achieved through the adoption of more open data standards (such as semantic web technologies).[22] [23] Indeed, the markup language XML and several of its extensions have been used to describe legal texts. In particular, the markup language LegalRuleML has

[18] B. Chau and M. Livermore, 'Computational Legal Studies Comes of Age' (2024) 1(1) European Journal of Empirical Legal Studies, 89 https://doi.org/10.62355/ejels.19684 [accessed 14 November 2024].
[19] T. Hvitved, 'Contract Formalisation and Modular Implementation of Domain-specific Languages' (PhD thesis, University of Copenhagen 2011), (2012) https://di.ku.dk/english/research/phd/phd-theses/2011/hvitved12phd.pdf [accessed 14 November 2024].
[20] S. Yuill, 'Section Editorial: Critical Approaches to Computational Law' (2019) Computational Culture 7 online at http://computationalculture.net/section-editorial-critical-approaches-to-computational-law/ [accessed 14 November 2024].
[21] H. Surden, 'Computable Contracts' (2012) 46/629 UC Davis Law Review 629, online at https://ssrn.com/abstract=2216866 [accessed 14 November 2024].
[22] Cummins (n 1).
[23] Cummins and Clack (n 3).

been developed for this purpose by the information technology standards consortium OASIS (www.oasis-open.org).[24]

InsurLE is compatible with, and can benefit from such modularisation approaches, because modularisation provides a higher-level structure within which InsurLE codings can be embedded.

Codified representations express the implicit logic of a contract wording explicitly as 'code' in a general-purpose programming language, such as Python or Prolog, or in a domain-specific computer language, such as Solidity, Lexon[25] or L4.[26] One of the first large-scale applications of this approach was the implementation of a large portion of the British Nationality Act in Prolog.[27] According to a review of the field of Logic and Law,[28] this work was "hugely influential for the development of computational representations of legislation, showing how logic programming enables intuitively appealing representations that can be directly deployed to generate automatic inferences". More recently, the application of LP to law has become a major contributor to the development of 'Rules as Code',[29] which is a code-first approach to drafting legal rules. The application of LP to the computational representation of insurance contracts has more recently been further advanced by the Codex Insurance Initiative at Stanford University.[30] [31] [32] [33]

A related, but logically distinct approach, in the area of smart contracts, involves the coding of reactive rules of the form *if X then do Y*, which generate actions *Y* in response to external events *X*.[34]

---

[24] T. Athan, G. Governatori, M. Palmirani, A. Paschke and A. Wyner, 'LegalRuleML: Design principles and foundations' (Reasoning Web. Web Logic Rules: 11th International Summer School 2015, Berlin, Germany, Tutorial Lectures 11, 151 31st July – 4th August 2015).
[25] H. Diedrich, *Lexon Bible: hitchhiker's guide to digital contracts* (3rd edn. Wildfire Publishing, London UK, 2020).
[26] G. Governatori and M. Wong, 'Defeasible semantics for L4' (POPL ProLaLa 2023. Centre for Computational Law Springer, 2023) https://ink.library.smu.edu.sg/cclaw/5 [accessed 14 November 2024].
[27] M. Sergot, F. Sadri, R. Kowalski, F. Kriwaczek, P. S. Hammond and H. T. Cory, 'The British Nationality Act as a logic program' (1986) 29(5) Communications of The ACM, 370 https://doi.org/10.1145/5689.5920 [accessed 14 November 2024].
[28] H. Prakken and G. Sartor, 'Law and logic: A review from an argumentation perspective' (2015) 227 Artificial Intelligence 214 https://doi.org/10.1016/j.artint.2015.06.005 [accessed 14 November 2024].
[29] M. Waddington, 'Rules As Code: Drawing Out the Logic of Legislation for Drafters and Computers' (2022) SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4299375 [accessed 14 November 2024].
[30] Genesereth (n 4).
[31] Genesereth (n 5).
[32] P. Carlson and M. Genesereth, 'Insurance Portfolio Analysis as Containment Testing' (2023) Frontiers in artificial intelligence and applications https://doi.org/10.3233/faia230957 [accessed 14 November 2024].
[33] O. R. Goodenough and P. J. Carlson, 'Words or code first? Is the legacy document or a code statement the better starting point for complexity-reducing legal automation?' (2024) 382/2270 Philosophical Transactions - Royal Society. Mathematical, Physical and Engineering Sciences https://doi.org/10.1098/rsta.2023.0160 [accessed 14 November 2024].
[34] M. Goldby, C. Reed, M. MacDonald, K. Richards and L. Stanbrough, *Triggering Innovation: How Smart Contracts Bring Policies to Life* (Lloyd's, 2019) online at https://assets.lloyds.com/assets/pdf-triggering-innovation-how-smart-contracts-bring-policies-to-life/1/pdf-triggering-innovation-how-smart-contracts-bring-policies-to-life.pdf [accessed 14 November 2024].

Although many of these codified representations have been written in logic programming languages, such as Prolog, they can often be translated into natural language, because of the close relationship between the logic which underpins LP and the logic of natural language. Moreover, the implementation of InsurLE translates wordings written in InsurLE into Prolog code which is very close to hand-coded Prolog representations. There are also variants of LE which include language features for expressing reactive rules in smart contracts.[35]

Large language models (LLMs), in contrast with the other approaches above, do not expose the modular or logical structure of a document. However, they do provide the ability for a user to input a query about an unstructured legal text, and to obtain an output, which may be an answer to the query.[36] Unfortunately, such answers are often incorrect, and even when the answers are correct, such systems generally cannot provide trustworthy explanations for their answers.[37]

Despite these current limitations, LLMs are improving at a rapid rate, especially in combination with Prolog[38] and are likely to provide increasingly more useful applications for computable contracts in the future. In particular, they are likely to prove useful for helping to translate existing natural language documents and queries into more structured, codified representations, including those written in InsurLE.

In summary, InsurLE combines the natural language expression of contracts of natural language processing systems with the logical and computational capabilities of codified representations. Moreover, it is compatible with the use of modularisation techniques, to provide a more highly structured representation of insurance contracts and of legal texts more generally.

---

[35] Kowalski (n 9).

[36] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, 'Language models are few-shot learners' in H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (eds), *Advances in Neural Information Processing of the Thirty-Second AAAI Conference on Artificial Systems*, volume 33, (Curran Associates, Inc. 1877, 2020) online at https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf [accessed 14 November 2024].

[37] G. Hill, M. Waddington and L Qiu, 'From pen to algorithm: optimizing legislation for the future with artificial intelligence' (2024) AI & Soc https://doi.org/10.1007/s00146-024-02062-3 [accessed 14 November 2024].

[38] S. Shchegrikovich, 'Use Prolog to improve LLM's reasoning' (2024) online at https://shchegrikovich.substack.com/p/use-prolog-to-improve-llms-reasoning [accessed 17 November 2024].

3. Introduction to the Nature and Structure of Insurance Contracts

Before introducing InsurLE in detail in later sections of this paper, we first present in this section a preview of InsurLE in the context of its application to the different kinds of clauses found in insurance contracts.

In the non-life context, an insurance contract represents two promises: (1) a promise made by an insurer to the insured (or policyholder) to indemnify the loss incurred by the insured in the event of an insured peril, and (2) a promise by the insured to pay a premium. Insurance policies often express this indemnification of a loss (which can be of different kinds, as specified by the policy) as the provision of coverage for 'costs in respect of the loss' to signal that the policy pays financial compensation, and only in exceptional and expressly noted circumstances provides for replacement or reinstatement of property. The insured's costs may include the value of insured property (in first party insurance) and the amount of damages associated with a claim, the legal representation costs for appealing against the legal liability for a damage, and other 'claims costs' (in third party or liability insurance).

At the heart of an insurance contract lies the ***insuring clause***, which describes the core coverage offered to the insured. Here is a simplified example:

> we will cover any cost
> which is in respect of any damage
> which is caused by a storm
> if the storm occurs during the period of insurance.

The scope of coverage expressed by the insuring clause is often extended with ***coverage extensions***, which are additional clauses in the contract, such as:

> we will cover any cost
> which is in respect of any damage
> which is caused by a burst pipe
> if the burst pipe occurs during the period of insurance.

These coverage extensions may be offered as standard, or as options, in which case their selection may result in an increase in the premium.

Most insurance contracts also contain ***exclusions***, which specify losses or costs of losses that are not covered by the contract. These exclusions can also be understood as exceptions to coverages. For example:

> we will not cover any cost
> which is in respect of any damage
> which is caused by wear and tear or negligence.

Exclusions may also contain ***exemptions.*** These exemptions can be understood as coverages that are excluded at first, but then reinstated (also known as 'writebacks'). Exemptions can also be understood as exceptions to exclusions. For example:

we will not cover any cost
which is in respect of any damage
which is caused by a loss of electricity
which occurs during a storm
unless the loss of electricity lasts for more than an hour.

The combination of an exclusion and an exemption is often stated in an alternative form, which gives emphasis to the exemption (as a coverage), rather than to the exclusion:

we will cover any cost
which is in respect of any damage
which is caused by a loss of electricity
which occurs during a storm
only if the loss of electricity lasts for more than an hour.

Coverages expressed by the insuring clause or coverage extension clauses are normally subject to ***general conditions***. Breach of or non-compliance with any of these conditions entitles the insurer to decline liability for a claim (unless it can be shown that a breach had already been remedied at the time of the loss or the 'non-compliance could not have increased the risk of the loss which actually occurred').[39]

General conditions have the same logical effect as exclusions. For example, the general condition that all electrical work must be certified to be safe can be expressed as an exclusion, i.e. as a non-coverage statement:

we will not cover any cost
if you have any electrical work carried out on the insured property
and it is not the case that
    the electrical work is certified to be safe.

All of the example clauses above are expressed in the syntax of InsureLE.[40] In particular, they illustrate one of the most distinguishing features of InsurLE, namely the use of restrictive relative clauses, signalled by the relative pronoun 'which'.[41]

[39] Insurance Act 2015, ss10 and 11.
[40] In many insurance contracts it is common to express general conditions as obligations, using a modal auxiliary verb such as 'must'. For example, the general condition above might be expressed as 'If you have any electrical work carried out on the insured property, then the electrical work ***must*** be certified safe'. Instead of expressing obligations in such modal terms, InsurLE expresses them in terms of the consequences of their violation. In the case of insurance contracts, violation of a general condition implies that the associated claim is not covered.
[41] Note that InsurLE syntax follows the British practice of using 'which' as a restrictive relative pronoun, in contrast with the American use of 'that'.

InsurLE limits relative clauses to restrictive clauses, and excludes non-restrictive clauses. Restrictive relative clauses logically add extra conditions to a rule, whereas non-restrictive relative clauses logically add extra conclusions to the rule. For example, the following sentence has two restrictive relative clauses:

> we will not cover any cost
> which is in respect of any damage
> which is caused by wear and tear or negligence.

The sentence is logically equivalent to a sentence with two conjoint conditions:

> we will not cover any cost
> if the cost is in respect of any damage
> and the damage is caused by wear and tear or negligence.

In contrast, the following sentence has one non-restrictive relative clause:

> we will cover a cost, which we will reimburse within one month of receiving a claim for the cost,
> if you are legally liable to pay the cost

The sentence is logically equivalent to a sentence with two conclusions[42]:

> we will cover a cost
> and we will reimburse the cost within one month of receiving a claim for the cost
> if you are legally liable to pay the cost.

InsurLE not only excludes such non-restrictive relative clauses, but it also excludes restrictive relative clauses which are embedded inside other clauses. For example, the following sentence has an embedded restrictive relative clause ‘which is with respect to a loss which occurs during the period of insurance':

> a cost which is in respect of a loss which occurs during the period of insurance
> occurs during the period of insurance

The sentence is logically equivalent to a sentence with the embedded clause written as a condition:

> a cost occurs during the period of insurance
> if the cost is in respect of a loss which occurs during the period of insurance.

These two exclusions (non-restrictive relative clauses and embedded restrictive relative clauses) eliminate from InsurLE the feature of centre-embedding, which has been shown to cause the greatest difficulty for

[42] To be more precise, the non-restrictive relative clause adds both an extra conclusion and an extra condition to the sentence. Stated as a separate sentence, it can be written in the form: ‘we will reimburse a cost within one month of a date if we receive a claim for the cost at the date and you are legally liable to pay the claim’.

reading comprehension in legal language.[43] They make InsurLE not only easier for computers to process, but in many cases they make InsurLE easier for lay people to read and understand.

At a higher level, the relationships between the different kinds of insurance clauses and their effect on the coverage provided can be understood in visual terms. as shown in Figure 1. Here the white and grey areas indicate coverage and non-coverage respectively.

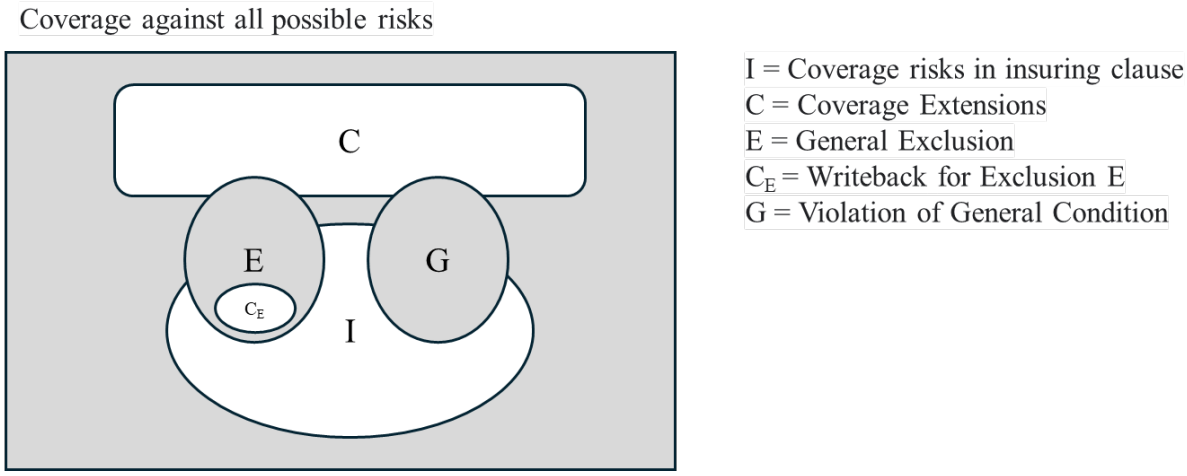

Figure 1. Visual representation of coverage offered by an insurance contract

At a still higher level, a typical insurance contract comprises two main objects: the ***schedule***, which serves as a summary of the coverage offered, and the ***wording***, which describes the coverage offered in detail. The main objectives and the forms of these objects are shown in Figure 2.

Contract Schedule	**Contract Wording**
Main Objectives • To provide a customer-specific overview of the coverage • To define optional coverage extensions • To define key limits (and sub-limits) for coverages • To identify any endorsements to the contract wording	**Main Objectives** • To provide details of the coverage • To specify defined terms • To define conditions for coverage • To define exclusions to coverage
Form • Document templates with embedded data • Data sent from Policy Admin System using XML or JSON • Schedules are sent to the customer in PDF format	**Form** • Fixed documents in PDF form • Contract wordings are often an expression of the 'product' • Wordings are sent to the customer in PDF format

Figure 2. The two principal insurance contract objects: the schedule and the wording.[44]

[43] Martínez et al. (n 17) 6.
[44] XML (Extensible Markup Language) and JSON (Javascript Object Notation) are computer languages that are used for the structuring, attribution and transfer of data.

The ***contract wording*** specifies the coverages offered to the insured together with any conditions, exclusions and exemptions. In essence, it is an expression of an ***insurance product*** using unstructured (but domain-specific) natural language, and can be over a hundred pages in length. Moreover, the wording is a fixed, singular object that is made available to customers in PDF format.

The ***contract schedule*** allows for the wording to be 'externally' customised to satisfy the specific requirements of customers. As indicated in Figure 2 above, the schedule confirms the broad area of coverage e.g. residential property, public liability, or professional indemnity. It also specifies the options selected from the optional coverage extensions specified in the wording and the ***limits of indemnity*** (the maximum amount payable under the contract) and sub-limits that apply to more specific areas of coverage (or individual coverages).

Endorsements are used to make changes to (or override) the wording. It is not uncommon for there to be several pages of endorsements, sometimes reaching a length similar to that of the wording itself. The data contained in the schedule are in a structured format (together with unique identifiers for the endorsements) and are stored on a PAS (Policy Administration System). We have not considered the treatment of endorsements in this paper.

For the work undertaken to develop InsurLE as an insurance-specific, controlled natural language, we have redrafted the wording of the 'Public and Product Liability' (PPL) Section,[45] which is part of an AXA Commercial general liability product. The PPL Section wording has the following four main sub-sections:

1. A subsection listing all of the ***defined terms***. This subsection specifies (and thereby) standardises the meaning of the defined terms used throughout the contractual documentation.
2. A coverage subsection that comprises the insuring clause (a statement of the core coverage offered) and both standard and optional extensions. This subsection also describes the limits of liability in more detail.
3. A general exclusion subsection that details what is not covered.
4. A general conditions subsection that details the conditions that must be adhered to by the insured for the promise of indemnification to be upheld by the insurer.

The InsurLE redrafting to the PPL codifies all of these sections, except for the section on defined terms, which is used to standardise the vocabulary of the codification.

[45] AXA (n 16).

4. A Brief Introduction to the Syntax of InsurLE

The main purpose of this section is to indicate the range of sentences that can be written in InsurLE, and to provide some guidance about the syntax of InsurLE for potential writers of InsurLE documents. Such guidance is not necessary for readers of InsurLE, because reading InsurLE requires no special training.

InsureLE is a restricted, unambiguous form of English. The lack of ambiguity ensures that all sentences expressed in InsurLE can be translated into unique logical representations, which enable trustworthy, logical reasoning.

All sentences in InsurLE are declarative or interrogative. Declarative sentences are used to represent both the wording of a contract and the data relevant to applying the contract in a particular use case or scenario. Interrogative sentences are used to pose a query or goal to the contract, given a scenario. InsurLE has no imperative or exclamatory sentences.

Declarative sentences are either ***facts***, which represent a property of an individual (or entity) or a relationship among individuals (or entities), or ***rules***, which consist of a conclusion and conditions. The components of a rule have the same syntactic form as facts, except they may contain variables, which stand for classes (or types) of individuals. Rules are used to represent the wordings of contracts, whereas facts are used to represent use cases or scenarios.

Facts are simple sentences about individuals which exist in the *universe of discourse* of some real or imaginary world. An **individual** in the universe of discourse can be a physical object, such as Bob's broken thumb, a conceptual object such as the specific insurance claim called claim 001, or a class of objects regarded as a single individual in its own right, like the class of all broken thumbs or the class of all bodily damages.

Facts can be true or false, depending on whether or not the properties or relationships which they assert are true in the world. For example, the following simple sentences are 'facts':

> Bob's broken thumb is a bodily injury.
> You are legally liable to pay claim 001.
> claim 001 is a cost.
> claim 001 is in respect of Bob's broken thumb.

Whether or not they are true facts depends on the world in the use case or scenario under consideration. In a typical insurance application, these facts would be given as input, and they might represent either a true fact, a hypothetical fact, a human judgement or even a counterfactual.

Here, 'Bob's broken thumb', 'bodily injury', 'you', 'claim 001' and 'cost' are noun phrases which *name* individuals in the universe of discourse. In formal logic such names of individuals are called *constants*, and

they are regarded as not having any meaningful internal structure. In symbolic logic, to assist this understanding of constants, they are normally represented by symbols, such as a, a1, a2, b, ... etc.[46]

An individual has a unique name, which is always written in the same way. The distinction between upper and lower case letters has no significance. To safeguard against possible ambiguity, in the current version of InsurLE, the pronouns 'he', 'she', 'they' and 'it' are not allowed to substitute for the names of previously mentioned individuals.

Here, the phrases 'is a', 'are legally liable to pay', 'is an' and 'is in respect of' *name* relations between individuals. In formal logic, such names of properties and relations are called *predicates*. Like constants, predicates are also regarded as not having any meaningful internal structure. Properties and relations also have unique names, which are written in the same way, wherever they occur, except for 'is a' and 'is an', which name the same relation, and which vary only to conform with ordinary English grammar.

To some extent, the distinction between predicates and constants is somewhat arbitrary. For example, the fact 'Bob's broken thumb is a bodily injury' could be understood either as expressing a property of Bob's broken thumb, or it could be understood as expressing a relationship between Bob' broken thumb and the class of bodily injuries. A writer of InsurLE (and of any computable, logical language) needs to make a choice between such alternative conceptualisations. In this case, the choice is made by means of the built-in 'is a' predicate in InsurLE. The choice can be visualised by means of a template:

_ is a _

where the underscore '_' indicates a slot that can be filled in either by a constant or by a variable. The choice to treat 'is a' as the name of a relation contrasts with the alternative choice of treating 'is a' as part of the name of a property. The alternative choice is less useful, because it would require many such templates for many similar properties, such as:

_ is a bodily injury
_ is a cost
_ is a loss,
etc.

There are three kinds of rules: (i) if-rules, (ii) expanded if-rules, which contain restrictive relative clauses, and (iii) only-if-rules.

(i) If-rules have the form *conclusion if conditions*, where *conclusion* is a simple sentence possibly containing variables, and *conditions* is a logical combination of simple sentences (using the logical connectives *and, or* and *it is not the case that.* For example:

[46] In addition to constants, Prolog also includes compound names with internal structure. For example, 'the king of France' is a compound name, which is an instance of the noun phrase 'the king of a country'. Noun phrases representing compound terms are not included in the current version of InsurLE.

we will cover a cost
if the cost is in respect of a damage
and the damage is caused by a burst pipe
and it is not the case that
    the damage is caused by wear and tear or negligence.

Here ‘a cost’, ‘a damage’ and ‘a burst pipe’ are variables; ‘the cost’ is the same variable as ‘a cost’ and ‘the damage’ is the same variable as ‘a damage’. In general, the first occurrence of a variable in a sentence is introduced by one of the determiners ‘a’, ‘an’ or ‘any’. Later occurrences of the same variable in the same sentence are represented by the same string of words, but with the determiner ‘a’, ‘an’ or ‘any' replaced by the determiner ‘the’.

A sentence containing variables is shorthand for all the sentences obtained by replacing (or ‘instantiating’) variables by constants. For example, the sentence above stands for such instances as:

we will cover claim 001
if claim 001 is in respect of my ruined kitchen worktop
and my ruined kitchen worktop is caused by burst pipe event 001
and it is not the case that
    my ruined kitchen worktop is caused by wear and tear or negligence.[47]

Here, the constant ‘claim 001’ replaces all occurrences of the variable ‘a cost’, ‘my ruined kitchen worktop’ replaces all occurrences of ‘a damage’ and ‘burst pipe event 001’ replaces the single occurrence of ‘a burst pipe’. The constant ‘wear and tear or negligence’ is to be understood as an unstructured constant, in which the words ‘and’ and ‘or’ are not to be understood as logical connectives.

Typed Variables. If-rules in InsurLE are identical to if-rules in LE, except that variables in InsurLE are typed. The types of variables in InsurLE are used to restrict the constants which can meaningfully instantiate the variables. For example, they disallow such syntactically correct, but meaningless instances of the rule above as:

we will cover jungle book
if jungle book is in respect of Rudyard Kipling
and Rudyard Kipling is caused by snow
and it is not the case that
    Rudyard Kipling is caused by wear and tear or negligence.

Here the constant ‘jungle book’ replaces all occurrences of the variable ‘a cost’, ‘Rudyard Kipling’ replaces all occurrences of ‘a damage’ and ‘snow’ replaces the single occurrence of ‘a burst pipe’. These

[47] If my ruined kitchen worktop is caused both by the burst pipe event 001 and by wear and tear or negligence, and if these two causes are represented by the corresponding facts in the input scenario, then it logically follows that it is not the case that we will cover claim 001.

instantiations of variables do not respect the intended interpretation of the types of the variables, assuming that it is not the case that 'jungle book is a cost', 'Rudyard Kipling is a damage' and 'snow is a burst pipe'.

In general, a *typed variable* in InsurLE is a noun phrase whose first word is one of the determiners 'a', 'an', 'any' or 'the'.[48] The remainder of the noun phrase is the *type* of the variable. For example, 'cost' is the type of the variable 'a cost' and 'burst pipe' is the type of the variable 'a burst pipe'.

Types in InsurLE impose a hierarchical structure on the universe of discourse. In this hierarchy, concrete objects are situated at the bottom of the hierarchy, and abstract classes appear at higher levels, with subclasses occurring directly below their parent class. This hierarchy of objects and classes can be viewed as part of a domain-specific ontology for insurance contracts[49].

The syntax of InsurLE uses the types of variables to represent this hierarchy of objects and classes, and to restrict the constants which can validly instantiate variables. So, for example, given the facts:

> my ruined kitchen worktop is a property damage.
> property damage is a damage.

the constant 'my ruined kitchen worktop' is a valid instance of the variable 'a damage'. This is because the conclusion 'my ruined kitchen worktop is a damage' follows logically from the given facts, together with the built-in untyped rule:

> X is a Y if X is a Z and Z is a Y.

Typed variables are implemented in InsurLE by replacing all occurrences of a typed variable in a rule by an untyped variable and by adding an extra condition to the rule expressing that the untyped variable is an instance of the type. So, for example, the InsurLE rule with typed variables:

> we will cover a cost
> if the cost is in respect of a damage
> and the damage is caused by a burst pipe

is rewritten in the implementation of InsurLE as the untyped LE rule:

> we will cover a X
> if X is a cost
> and X is in respect of a Y
> and Y is a damage
> and Y is caused by a Z

[48] However, if a noun phrase beginning with one of the determiners 'a' or 'an' is preceded by the word 'is', then the noun phrase does not represent a variable. Instead, the determiner is part of the 'is a' or 'is an' predicate. For example, 'a cost' is a variable in 'we will cover a cost', but 'a cost' is not a variable in 'claim 001 is a cost'.

[49] An *ontology* is a set of concepts, properties and relationships in a subject area or domain underpinning the vocabulary of a language for that domain.

and Z is a burst pipe.

Here X, Y and Z are untyped variables, which can be instantiated by arbitrary constants without any restrictions. The intended type restrictions are imposed by the conditions ‘X is a cost’, ‘ Y is a damage’ and ‘Z is a burst pipe’.

(ii) Expanded if-rules in InsurLE provide the option of incorporating selected conditions of an if-rule into the conclusion of the rule as restrictive relative clauses. This expansion of the conclusions of rules by means of restrictive relative clauses helps to make the rules more informative, and helps to make them to more closely resemble typical insurance writing style. For example, the if-rule above can be written as the expanded if-rule:

we will cover a cost
which is in respect of a damage
which is caused by a burst pipe
if it is not the case that
the damage is caused by wear and tear or negligence.

In general, a restrictive relative clause in an expanded if-rule is a simple, fact-like sentence containing a variable, with the relative pronoun ‘which’ replacing the variable and positioned at the front of the sentence.

Logically, restrictive relative clauses have the same meaning as conditions of rules. In fact, the implementation of InsurLE, in effect, translates restrictive relative clauses into conditions. For example, the expanded rule:

we will cover a cost
which you are legally liable to pay.

has the same meaning as the rule:

we will cover a cost
if you are legally liable to pay the cost.

Here, the condition of the rule has been written as a restrictive relative clause, by deleting the variable ‘the cost’ from the end of the condition, and adding ‘which’ at the beginning.

The relative pronoun ‘which’ at the beginning of a restrictive relative clause in an expanded if-rule refers to the variable that immediately precedes the pronoun. So that ‘which’ in ‘we will cover a cost which you are legally liable to pay’ refers to the variable ‘a cost’ which immediately precedes ‘which’. This constraint, that a relative pronoun always refers to the variable that immediately precedes it, eliminates any potential ambiguity about the referent of the pronoun, and it helps to ensure that relative clauses in InsurLE are understood in the same way both by humans and by computers.

(iii) Only-if rules have the same syntax as expanded if-rules, but with ***if*** replaced by ***only if***. For example:

we will cover a cost
which is in respect of a damage
only if it is not the case that
    the damage is caused by wear and tear or negligence.

Logically, this has the same meaning as the sentence:

it is not the case that we will cover a cost
which is in respect of a damage
if the damage is caused by wear and tear or negligence.

The implementation of InsurLE uses this equivalence of meaning to translate only-if rules into ordinary if-rules in LE. However, to facilitate the representation of rules and exceptions, it replaces the negative conclusion 'it is not the case that we will cover a cost' by a positive, but contrary conclusion[50] 'we will not cover a cost'.

Rules and Exceptions in InsureLE. In LE, a rule which admits an exception is written with a negative condition, which anticipates and excludes the exception. For example, the negative condition in the following rule excludes costs of damages caused by wear and tear or negligence:

we will cover a cost
which is in respect of a damage
which is caused by a burst pipe
if it is not the case that
the damage is caused by wear and tear or negligence.

However, when there are possibly several exceptions to a general rule, it is better to write the negative condition in a more general form, so that the rule as a whole has the general form *a conclusion holds if certain conditions hold and it is not the case that the contrary of the conclusion holds*. For example, if the rule above has two exceptions, one for damage caused by wear and tear or negligence, as above, and one for the cost of the damage exceeding the limit of indemnity, then it is better to write the rule and its exceptions in the more general form:

we will cover a cost
which is in respect of a damage
which is caused by a burst pipe,
if it is not the case that
we will not cover the cost.

[50] In the same way that 'bad' is the positive contrary of 'good', 'guilty' is the positive contrary of 'innocent' and 'uncool' is the positive contrary of 'cool', the predicate 'we will not cover' is the positive contrary of the predicate 'we will cover'. In particular, the word 'not' in 'we will not cover' is not treated as a logical operation, in the same sense that 'un' is not a logical operation in 'uncool'.

we will not cover a cost
which is in respect of a damage
if the damage is caused by wear and tear or negligence.

we will not cover a cost
if the cost exceeds the limit of indemnity.

The inclusion in LE of an explicit negative condition, to exclude possible exceptions, clarifies the logical relationship between a general rule and its exceptions. However, in ordinary natural language, it is common to omit such negative conditions, and to take for granted that a general rule can implicitly have exceptions which override the general rule. So, instead of stating a rule in the form *a conclusion holds if certain conditions hold and it is not the case that the contrary of the conclusion holds* it is more common to omit the negative condition, and to state the rule in the simpler form *a conclusion holds if certain conditions hold*. But, in such a case, when there are potentially conflicting rules of the simplified form *a conclusion holds if certain conditions hold* and *the contrary conclusion holds if certain other conditions hold*, it is important to indicate which sentence states the general rule and which sentence states an exception. In ordinary English, it is common to make this indication, simply by writing the exceptions after the general rule. For example, *a conclusion holds if certain conditions hold. The contrary conclusion holds if certain other conditions hold*.

InsurLE follows the writing practice of ordinary natural language in the insurance industry by omitting explicit reference to possible exceptions in the coverage rules themselves, with the convention that the relevant exceptions are expressed separately in non-coverage rules, which typically are written after the coverage rules. So, for example, the coverage rule for damage caused by a burst pipe can be written in InsurLE in the simplified, form:

we will cover a cost
which is in respect of a damage
which is caused by a burst pipe.

InsurLE implements reasoning with such simplified rules and exceptions by adding explicit negative conditions of the form *it is not the case that the contrary of the conclusion holds* to the general rules, without adding corresponding negative conditions to the exceptions.

Interrogative sentences, also called queries, in InsurLE have the same syntax as the conditions of rules, except that variables which have only a single occurrence in the query can optionally be replaced by the interrogative pronoun ‘which’. For example:

we will cover which cost?
which cost is in respect of which damage?

which cost is in respect of a damage which is caused by a burst pipe?

Here in the first two queries, all occurrences of 'which' are interrogative pronouns. But in the third query, the first occurrence of 'which' is an interrogative pronoun; the second occurrence is a restrictive relative pronoun.

5. The Semantics of InsurLE

The meaning of a set of declarative sentences in InsureLE is identical to the meaning obtained by translating the sentences into LE, and by interpreting the resulting LE sentences as a logic program:

- The translation rewrites an InsurLE rule of the form *we will cover a cost if certain conditions hold* as an LE rule of the form *we will cover a cost if certain conditions hold and it is not the case that we will not cover the cost*.
- It rewrites an only-if-rule of the form *we will cover a cost only if certain conditions hold* as an LE rule of the form *we will not cover a cost if it is not the case that certain conditions hold*.
- It rewrites a restrictive relative clause in a rule as a condition of the rule.
- It rewrites a typed variable in a rule as an untyped variable, and it adds an extra condition to the rule restricting the untyped variable to its type. It uses the built-in rule *X is a Y if X is a Z and Z is a Y* behind the scenes, to evaluate whether or not the extra condition holds.

An example of this rewriting is illustrated in Figure 3.

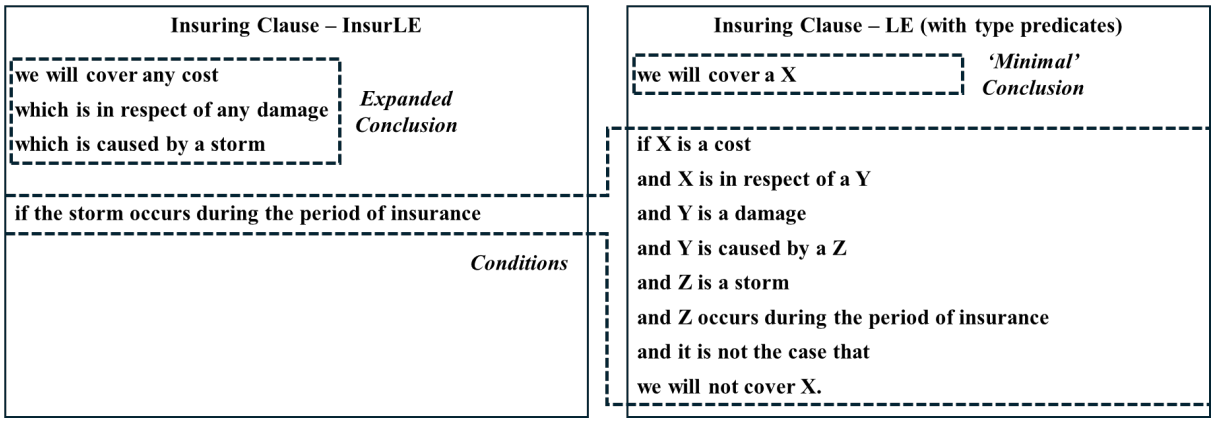

Figure 3. An insuring clause in InsurLE and its representation in LE.

Clearly, the LE representation in Figure 3 is much more cumbersome than the InsurLE representation. The LE representation translates the typed variables 'any cost', 'any damage' and 'a storm' in InsurLE into meaningless symbolic variables X, Y and Z, and it associates the variables with their appropriate types by means of 'is a' conditions.

However, the advantage of the LE representation is the simplicity of its semantics, which is based on the semantics of classical predicate logic, but employs a *database*[51] restriction on the possible interpretations of sentences. It restricts attention to interpretations in which:

1. The only individuals are ones that have names in the language of the logic program.
2. Every individual has a unique name, in the sense that different names refer to different individuals.
3. The only facts that are true in the interpretation are the facts that are known to be true given the facts and rules in the program.

The first two restrictions, in effect, eliminate the distinction between individuals and their names. The third restriction[52] eliminates the distinction between the facts that are true in the world and the facts that can be derived from the program. Taken together, the three restrictions view a logic program as defining all the facts that are true in the world, and these facts are said to be the *intended interpretation* of the program. Given a logic program, written in a language with a fixed vocabulary of constants and predicates:[53]

- A fact is true in the *intended interpretation* of the program

  if the fact is a fact in the logic program

  or the fact is the conclusion of a variable-free instance of a rule in the program

  and the conditions of the variable-free instance of the rule are true

  in the *intended interpretation* of the program.

- No other facts are true in the *intended interpretation* of the program.

Complications can arise with this definition if the truth of a fact which is an instance of the conclusion of a rule depends on the negation of the truth of the same fact either directly in the same instance of the rule, or indirectly in other instances of the rules. Programs which do not suffer from this complication are said to be

[51] S. Russell and P. Norvig, *Artificial intelligence: A Modern approach* (4th ed. Prentice Hall, 2021).
[52] This restriction is sometimes called the 'closed-world assumption': R. Reiter, 'On closed world databases' in B. L. Webber and N. J. Nilsson (eds), *Readings in artificial intelligence*, (Morgan Kaufmann, 2021), 119. The closed world assumption can also be understood as interpreting a set of if-rules with the same conclusion but with alternative conditions as a definition that the conclusion holds ***if and only if*** one of the alternative conditions holds. K. L. Clark, 'Negation as Failure' in: Gallaire, H., Minker, J. (eds) *Logic and Databases* (Springer, Boston, MA, 1978), https://doi.org/10.1007/978-1-4684-3384-5_11 [accessed 14 November 2024]).
[53] Notice that this definition of the semantics of LP is itself written informally in LP form, using LP as a *meta-language* to define the semantics of LP. The last sentence expresses the closed world assumption. This use of LP as a meta-language also underpins the use of Prolog to implement meta-reasoners, which define the reasoning strategies of other reasoners.

stratified[54] and for such stratified programs, the definition above specifies a single intended interpretation, which encapsulates the meaning of the program. Fortunately, the logic programs that result from rewriting a set of facts and rules in InsurLE are always stratified, thanks in part to the way in which coverage depends upon noncoverage for possible exceptions, but noncoverage does not depend upon coverage.

Given any stratified logic program, an **answer** to a query is any instance of the query which is true in the intended interpretation of the program. Consider, for example, the InsurLE rules and the corresponding LE rules (with stratified negation) in Figure 4.

Rules with typed variables in InsurLE	**Corresponding rules with untyped variables in LE**
we will cover a cost **which is in respect of any damage** **which is caused by a burst pipe.**	**we will cover a X** **if X is a cost** **and X is in respect of a Y** **and Y is a damage** **and Y is caused by a Z** **and Z is a burst pipe** **and it is not the case that we will not cover X.**
we will not cover a cost **which is in respect of any damage** **if the damage is caused by wear and tear or negligence.**	**we will not cover a X** **if X is a cost** **and X is in respect of a Y** **and Y is a damage** **and Y is caused by wear and tear or negligence.**

Figure 4. Rules with typed variables in InsurLE and the corresponding rules in LE.

Consider the following scenario:

> claim 1 is in respect of destroyed living room carpet.
> claim 2 is in respect of warped and water-stained wooden kitchen worktop.
> destroyed living room carpet is caused by burst pipe event 001.
> warped and water-stained wooden kitchen worktop is caused by wear and tear or negligence.
> claim 1 is a cost.
> claim 2 is a cost.
> destroyed living room carpet is a damage.
> warped and water-stained wooden kitchen worktop is a damage.
> burst pipe event 001 is a burst pipe.

Given the scenario and the two rules in Figure 4, the intended interpretation assigns truth to all the facts in the scenario and to the two additional facts:

[54] K.R. Apt, H.A. Blair and A. Walker, Towards a Theory of Declarative Knowledge (1988) In *Foundations of Deductive Databases and Logic Programming*, Minker, J., ed. Morgan Kaufmann Publishers Inc., 89.

we will cover claim 1.

we will not cover claim 2.

No other facts that can be expressed in the vocabulary are true. The first of these two additional facts is the only answer to the query: we will cover which thing?[55]

6. Computation in InsurLE

In the same way that the semantics of InsurLE is inherited from the semantics of logic programming, computation in InsurLE is inherited from computation in logic programming. Computation in InsureLE is performed by first translating the given InsurLE facts, rules and query into a logic program and query, and then employing a logic programming (LP) reasoner to search for answers to the query. In our experiments, reasoning is performed by Prolog. But computation in InsurLE can also be performed by other logic programming reasoners, such as s(CASP).[56] The translators themselves are also implemented in Prolog.

In the Datalog family of LP languages, most implementations reason forwards (or bottom-up) from the given facts, using rules to derive new facts, until enough facts have been derived to answer the query directly. This way of reasoning mirrors the natural understanding of the definition of the intended interpretation of logic programs: A new fact is derived as an instance of the conclusion of a rule, by matching the positive conditions of the rule with existing facts and by showing that the negated conditions of the rule match no existing facts. Stratification ensures that if a negated condition matches no existing fact then it will not match any new facts in the future.

In contrast, in Prolog, reasoning proceeds in the opposite direction, backwards (or top-down) from a query (or goal), reducing goals that unify[57] with the conclusions of rules to subgoals that correspond to the conditions of the rules. Positive subgoals are solved either by unifying them with facts or by unifying them with the conclusions of the same or other rules, and reducing those subgoals to further subgoals, until all subgoals are eventually solved by facts. Negative subgoals of the form *it is not the case that p* are solved by negation as failure,[58] showing that the negated goal (or query) *p* cannot be solved, using the same backward reasoning, goal-reduction method.

---

[55] However, if the negative condition ‘it is not the case that we will cover X’ is added to the second rule, then the negative conditions would not be stratified, and there would be two interpretations. In both interpretations ‘we will cover claim 1’ is true. But in one interpretation ‘we will cover claim 2’ is true, and in the other interpretation ‘‘we will cover claim 2’ is false.

[56] G. Sartor, J. Dávila, M. Billi, G. Contissa, G. Pisano and R. Kowalski, ‘Integration of Logical English and s(CASP)’ (2022) Aachen : CEUR-WS.org Sun SITE Central Europe / RWTH Aachen University, online at https://hdl.handle.net/11585/919580 [accessed 14 November 2024].

[57] Unification takes two simple sentences and instantiates the variables, making the resulting two instantiated sentences identical.

[58] K. L. Clark, ‘Negation as Failure’ in: Gallaire, H., Minker, J. (eds) *Logic and Databases*. (Boston, MA, Springer 1978). https://doi.org/10.1007/978-1-4684-3384-5_11 [accessed 14 November 2024].

In the Answer Set Programming (ASP) variant of LP, negation need not be stratified, and reasoning does not involve any query at all, but the reasoner generates all of the facts that are true in some interpretation, as a consequence of the given rules and the facts.

The current implementation of InsureLE has two modes of execution, both of which reason with the Prolog program into which the InsurLE rules have first been translated. The simplest mode of execution uses Prolog itself to answer queries. The more elaborate mode uses a *meta-reasoner*[59] written in Prolog.[60] The meta-reasoner also performs Prolog-style backward reasoning, but with added features, such as generating conditional answers and generating explanations for answers. For example, consider the InsurLE rules in Figure 4 and the following subset of the facts given in the previous scenario:

> claim 1 is in respect of destroyed living room carpet.
> claim 2 is in respect of warped and water-stained wooden kitchen worktop.
> warped and water-stained wooden kitchen worktop is caused by wear and tear or negligence.
> claim 1 is a cost.
> claim 2 is a cost.
> destroyed living room carpet is a damage.
> warped and water-stained wooden kitchen worktop is a damage.

Given the same query as before:

> we will cover which cost?

the InsurLE reasoner generates the conditional answer:

> we will cover claim 1
> if destroyed living room carpet is caused by a burst pipe
> and it is not the case that
> destroyed living room carpet is caused by wear and tear or negligence.

The reasoning is performed by translating the InsurLE rules, facts and query into Prolog, and then using the Prolog meta-reasoner to reason backwards from the query. The meta-reasoner unifies the query with the conclusion of the first rule, and it proceeds to evaluate the positive conditions of the rule against the given facts. The first three conditions have two answers, one where X is ‘claim 1’ and Y is ‘destroyed living room carpet’, and the other where X is ‘claim 2’ and Y is ‘water-stained wooden kitchen worktop’.

---

[59] S. Costantini, ‘Meta-reasoning: A survey’ in A. C. Kakas and F. Sabri (eds), *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski, Part II* (Springer, Berlin Heidelberg, 2002), 253.

[60] Meta-reasoners are implemented in Prolog by using Prolog as a meta-language to define the reasoning strategy of another logic. In the case of our meta-reasoner, the other logic and reasoning strategy are Prolog-like. But for the sake of providing explanations and other useful features, the meta-reasoner records details of proofs, which the Prolog reasoner discards or ignores.

In the case where X is ‘claim 1’, there are no facts that unify with the third and fourth positive conditions, ‘destroyed living room carpet is caused by a Z’ and ‘Z is a burst pipe’, which then become candidate conditions for a conditional answer. It remains to evaluate the negative condition ‘it is not the case that we will not cover claim 1’. This is done by negation as failure, reasoning backwards from the conclusion of the second rule. In this case, the first three conditions of the second rule are evaluated to ‘true’. The fourth condition does not unify with any facts or conclusions of rules, and this condition can also contribute to a candidate condition of a conditional answer. Putting all the candidate conditions together, we obtain the answer, which can be written in LE as:

> we will cover claim1
> if destroyed living room carpet is caused by a Z
> and Z is a burst pipe
> and it is not the case that
> destroyed living room carpet is caused by wear and tear or negligence.

In the typed-variable syntax of InsurLE, the first two conditions are rewritten as the single condition ‘destroyed living room carpet is caused by a burst pipe’.

In the case where X is ‘claim 2’, all four conditions of the second rule evaluate to ‘true’, showing that the conclusion:

> we will not cover claim 2.

holds. As a result, the negative condition of the first rule fails to hold and the conclusion ‘we will cover claim 2’, fails to be another answer to the query.

Note that conditional answers relax the closed world assumption, by allowing for the possibility that some instances of the conditions of an answer might be true, even if they are not currently known to be true.[61]

7. The Insuring Clause of a General Liability Insurance Product in InsurLE

As already mentioned, we have rewritten the wording of a general liability insurance contract (the Public and Product Liability (PPL) Section from AXA Commercial in the UK) in InsurLE, and we have experimented with using a meta-reasoner written in Prolog, to generate conditional answers to queries, given incomplete scenarios. In this section, we present some examples of our experiment, using the insuring clause of the PPL, and its associated exclusion and exemption. Figure 5 compares the original PPL insuring clause with a representation of the insuring clause in InsureLE.

[61] This approach to generating conditional answers is based on the approach in R. Kowalski, F. Toni and G. Wetzel, ‘Executing suspended logic programs’ (1998) 34 Fundamenta Informaticae 203.

PPL Insuring Clause – Original Form

We will cover the amount of damages
which you are legally liable to pay
in respect of
1 bodily injury
2 personal injury
3 property damage
4 nuisance or trespass
occurring during the period of insurance
in connection with the business.

PPL Insuring Clause - InsurLE

We will cover any amount of damages
which you are legally liable to pay
and which is in respect of a loss
which is a
1 bodily injury or
2 personal injury or
3 property damage or
4 nuisance or
5 trespass
if the loss occurs during the period of insurance
and the loss occurs in connection with the business.

Figure 5. The PPL insuring clause in its original form and expressed in InsurLE.

Here, the relative pronoun 'which' in the relative clause 'which you are legally liable to pay' is potentially ambiguous. In theory, it could refer to one of the nouns 'amount' or 'damages', or it could refer to the entire noun phrase 'any amount of damages'. However, in the built-in InsurLE ontology, 'amount of damages' represents a single concept, which is a subtype of the type 'cost'. This relationship between subtype and type is represented by the built-in fact 'amount of damages is a cost'. In the context of this ontology, this occurrence of 'which' refers unambiguously to the variable 'any amount of damages' which immediately precedes it, and not to 'amount' or 'damages'.

As a general rule, the relative pronoun 'which' refers to a variable which immediately precedes it. However, if 'which' is preceded by 'and', as in the case of 'and which is in respect of a loss', then there must be a previous occurrence of 'which' in the same sentence, and the two occurrences of 'which' refer to the same variable. In this case, they both refer to the same variable 'any amount of damages'.

Notice that, compared with the original English wording 'the amount of damages', the InsurLE wording 'any amount of damages' uses the determiner 'any' rather than 'the', because 'any' introduces a variable, whereas 'the' refers to a variable that has already been introduced.[62]

One of the distinguishing features of InsurLE, like that of its parent, LE, is that it incorporates only a minimal knowledge of English grammar. In particular, the same predicate needs to be expressed in the same way in all of its occurrences. This explains why 'occurring' is replaced by 'occurs' and 'in connection with' is expanded into a full predicate 'occurs in connection with'.

As a matter of detail, the current implementation shows how several occurrences of the same expression (in this case 'the loss is a') can be 'factored out' of several similar conditions. Also, for the sake of uniformity, 'nuisance or trespass' has been rewritten as separate alternatives.

[62] In natural language, a variable can be introduced by the definite determiner 'the' if the value of the variable is a function of other variables in the sentence. We are exploring the inclusion of this feature in future extensions of InsurLE.

On the one hand, the comparison in Figure 5 shows how close the logically precise, computable InsureLE wording is to the original English wording. It shows, therefore, that, if the original English wording were actually written in InsureLE, then the fidelity of the computable coding to the original wording would be guaranteed simply by the fact that they are one and the same representation. On the other hand, it also shows some of the subtle differences between the two representations.

An exclusion to a general liability insuring clause in InsurLE. The coverage expressed in the PPL insuring clause has an exclusion for damage arising from the release or escape of pollutants, with an exception (or exemption) if the damage is caused by a sudden incident. The full clause is shown in the left column of Figure 6.

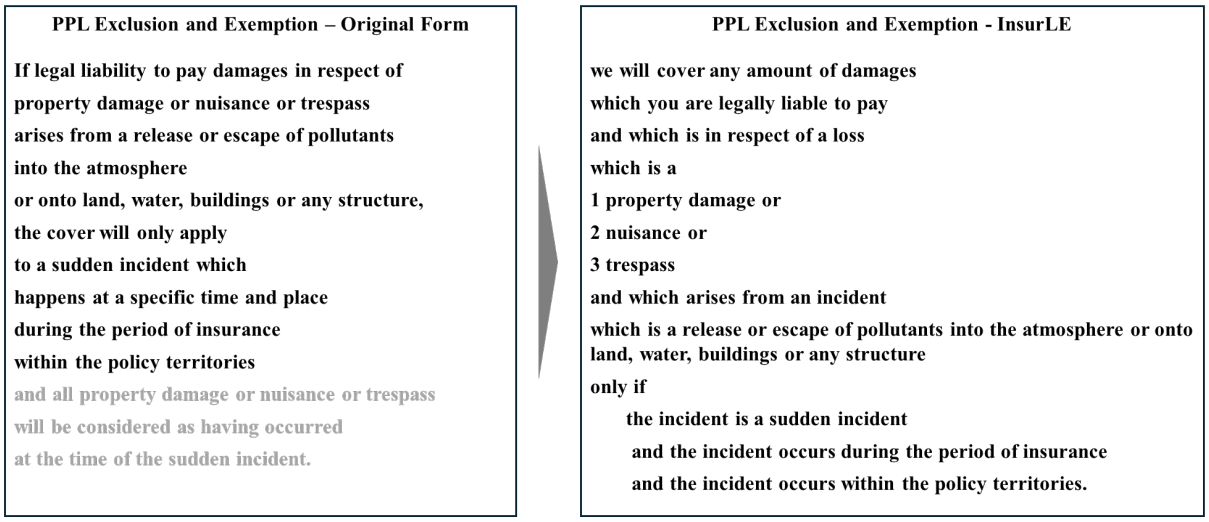

Figure 6. The exclusion and exemption of the PPL insuring clause in its original form and in InsurLE.

The full clause has two parts: The first part expresses the exclusion and exemption. The second part (greyed out in Figure 6) 'all property damage or nuisance or trespass will be considered as having occurred at the time of the sudden incident' is not a restrictive clause which modifies the meaning of the first part of the clause. It is a non-restrictive clause, which adds an extra conclusion to the first part of the clause.

The second part is needed, to determine whether, in the case of a sudden incident, the condition 'the loss occurs during the period of insurance' of the insuring clause applies. This can be represented simply in InsurLE by the rule:

a loss occurs during the period of insurance
if the loss arises from an incident
and the incident is a sudden incident
and the incident occurs during the period of insurance.

For example, if destroyed living room carpet arises from a sudden spillage of oil, which occurs during the period of insurance, then the destroyed living room carpet will be deemed to have also occurred during the

period of insurance, even if the damage to the carpet actually occurred after the period of insurance, as a delayed reaction to the spillage.

The right column of Figure 6 shows the InsurLE representation of the actual exclusion and exemption. The explicit use of the logical connective ***only if*** in InsurLE is more precise than the original English wording 'only applies'. This greater precision helps to ensure that the resulting InsurLE representation is both logically unambiguous and computable.

Notice also that to apply the InsurLE rule in Figure 6, it is necessary to determine whether a loss arises from a-'sudden incident', when the incident occurs and where it occurs. These judgements will typically be made by a loss adjuster, and either will be given as part of the input scenario, or will be generated as conditions of a conditional answer.

The InsurLE representation in Figure 6 follows the lead of the original wording, using 'only if' as the logical interpretation of 'only applies'. However, InsurLE also provides the writer with the flexibility to write the rule more explicitly as an exception to the coverage clause in Figure 5. The box on the right of Figure 7 shows this option. The two ways of writing the rule are logically equivalent.

PPL Exclusion and Exemption –
InsurLE only-if rule

we will cover any amount of damages
which you are legally liable to pay
and which is in respect of a loss
which is a
1 property damage or
2 nuisance or
3 trespass
and which arises from an incident
which is a release or escape of pollutants into the atmosphere or onto land, water, buildings or any structure
only if
    the incident is a sudden incident
    and the incident occurs during the period of insurance
    and the incident occurs within the policy territories.

PPL Exclusion and Exemption –
InsurLE extended if-rule

we will not cover any amount of damages
which you are legally liable to pay
and which is in respect of a loss
which is a
1 property damage or
2 nuisance or
3 trespass
and which arises from an incident
which is a release or escape of pollutants into the atmosphere or onto land, water, buildings or any structure
if it is not the case that
    the incident is a sudden incident
    and the incident occurs during the period of insurance
    and the incident occurs within the policy territories.

Figure 7. The exclusion and exemption of the PPL insuring clause as an only-if rule and as an extended if-rule in InsurLE.

8. An Example Use Case of InsurLE

Figure 8 illustrates the use of InsureLE to provide a conditional answer to a query, given incomplete information.[63] It shows how the answer can include both explicit conditions and conditions embedded in

[63] Note that, in order to obtain the answer in Figure 8, it is necessary to reason that 'Bob's broken thumb is a loss', using the input fact 'Bob's broken thumb is a bodily injury', the built-in fact 'bodily injury is a loss' and the built-in rule: 'X is a Y if X is a Z and Z is a Y'.

the answer as restrictive relative clauses. In this example, the distinction between the two kinds of conditions mirrors the way in which the conditions are represented in the insuring clause itself.

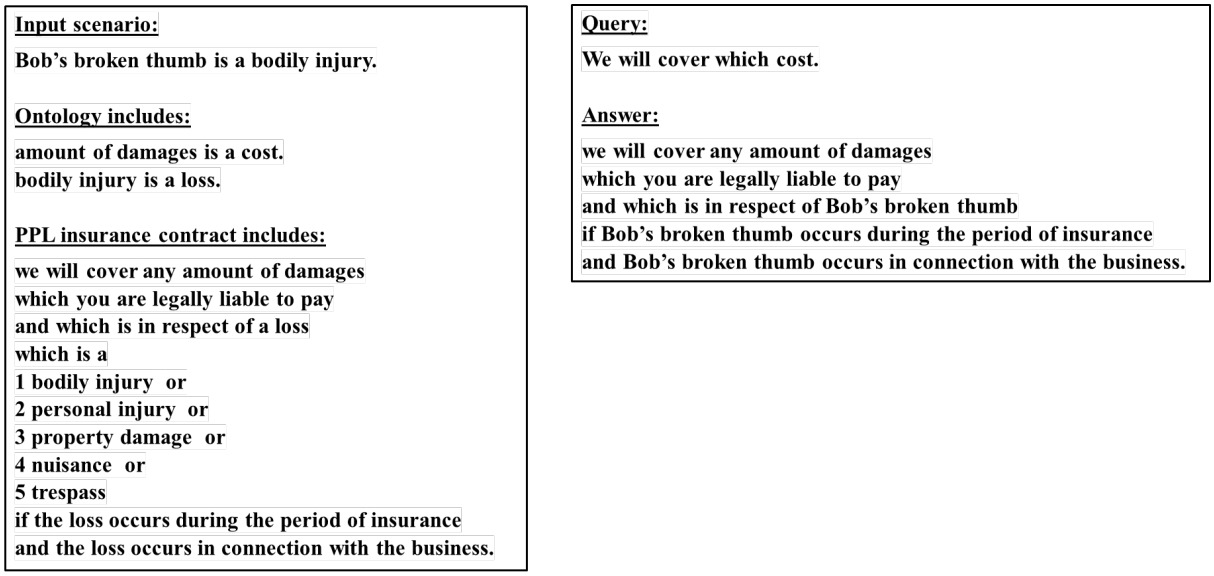

Figure 8. A conditional answer to a query with incomplete input.

The Figure suggests a typical application in which a claims handler might be provided with incomplete information about a claim, as in the box on the left in Figure 8. In the box on the right, the user inputs a query, and the reasoner outputs a conditional answer, which identifies additional information that needs to be provided for the claim to be successful. Additionally, a potential client could use the reasoner to obtain general information about coverage without having to provide complete information about a hypothetical scenario.

9. Conclusions and Further Work

In this paper, we presented the controlled natural language InsurLE, and we showed that it can serve simultaneously both as the natural language expression of a contract and as its codified representation. We argued that this combination of natural language and codified representation enables InsurLE to fulfil the promise of computable contracts and to do so in a way that respects and complements existing insurance industry practices.

In the first stage of our work, we translated the majority of the Public and Products Liability (PPL) wording from an AXA Commercial product into LE, and we demonstrated our preliminary results to several insurance professionals. Their feedback helped us to identify the extensions of the syntax of LE which we have incorporated into InsurLE. These extensions include:

- a more natural treatment of *typed variables*,

- the use of *restrictive relative clauses* in the conclusions of rules,
- the hiding of *unless the contrary conditions* and
- the use of *only if* for expressing the combination of exclusions and exemptions.

Most of these extensions have been investigated before, but we found that their selection and combination is particularly useful for representing wordings in the insurance domain. The first two of these extensions are already features of many other CNLs, including Attempto Controlled English (ACE)[64] [65] [66] and PENG,[67] which cover a much broader range of syntax than LE and InsurLE. In contrast, InsurLE aims to provide a minimal syntax, which maximises naturalness and expressive power, but avoids the syntactic complexities of traditional legalese. In particular, we have eliminated centre-embedded clauses, which have been found to inhibit readers' understanding and recall of legal texts to a greater degree than other features of legal contracts.[68]

It remains to be seen whether, because of its minimal, but expressive syntax, InsurLE and its further extensions might serve, not only as a CNL for insurance contracts, but also as a form of ***plain English***, which can be used to make legal texts more accessible to a wider range of readers with or without the assistance of computers.

The third extension, hiding *unless the contrary conditions,* uses the approaches of Kowalski and Sadri[69] and of Satoh et al.[70] to represent rules and exceptions in a form that is closer to their representation in natural language and to their expression in insurance contract wordings.

The fourth extension, using 'only if' to express the combination of exclusions and exemptions, is possibly the most original feature of InsurLE syntax, motivated by its frequent appearance in insurance contract wordings.

In addition to developing the syntax of InsurLE, we also tested the direct use of Prolog as the InsurLE reasoner. However, when we demonstrated the reasoner to our insurance colleagues, they observed that the reasoner was not adequate for typical applications in handling claims, where the given information about a

---

[64] N. E. Fuchs and R. Schwitter, 'Attempto Controlled English (ACE)' CLAW 96, The First International Workshop on Controlled Language Applications Katholieke Universiteit Leuven 1996).
[65] N. E. Fuchs, K. Kaljurand and T. Kuhn, 'Attempto Controlled English for Knowledge Representation' (2008) 5224 Reasoning Web. Lecture Notes in Computer Science 104. https://doi.org/10.1007/978-3-540-85658-0_3 [accessed 14 November 2024].
[66] N. E. Fuchs 'Attempto project', 2013 http://attempto.ifi.uze/ h.ch/sit [accessed 14 November 2024].
[67] R. Schwitter, 'English as a formal specification language' in *Proceedings: 13th International Workshop on Database and Expert Systems Applications* (IEEE, 2002).
[68] Martínez et al. (n 17) 6.
[69] R. Kowalski and F. Sadri, 'Logic programs with exceptions' (1991) 9 (3–4) New Generation Computing 387.
[70] K. Satoh, K. Asai, T. Kogawa, M. Kubota, M. Nakamura, Y. Nishigai, K. Shirakawa and C. Takano 'PROLEG: an implementation of the presupposed ultimate fact theory of Japanese civil code by PROLOG technology' in *JSAI international symposium on artificial intelligence*. (Springer, Berlin, Heidelberg, 2010).

particular case is often incomplete. We responded to this feedback by implementing a translation of InsurLE into s(CASP), and by using the s(CASP) reasoner to generate answers with missing information.[71] However, we then discovered complications with the resulting need to generate names of hypothetical individuals. This work is ongoing; and further improvements, such as the generation of conditional answers, still need to be implemented and tested.

In theory, the reasoner could also be extended to help a drafter to comply with legal requirements, including those imposed by regulators such as the UK Financial Conduct Authority (FCA). However, this capability would require the use of a more powerful reasoner, such as the one developed by Trevor Bench–Capon[72] for aiding policy makers to formulate legislation with the aid of logical models.

We are exploring further extensions of InsurLE, to make it closer to current insurance writing practice, while ensuring that it remains unambiguous and computable, and while avoiding syntactic features, such as centre-embedding, which make traditional legal contracts unnecessarily complicated and difficult to understand. Some of these extensions, such as rules with more than one conclusion, and rules and queries with restrictive relative clauses, have already been mentioned. In all of these cases, the semantics and computation of answers is obtained by translating the extensions into ordinary LP.

We also plan to extend our interactions with insurance professionals, to help ensure that the further development of InsurLE is relevant for the needs of the insurance industry. In particular, we are investigating which application areas are likely to be most useful.

One of the most pressing tasks for future work is to gain more practice with writing insurance contracts in LE, and to develop computer aids to help with drafting and redrafting contracts. We are already using a VS Code predictive text editor for LE, which needs to be extended for InsurLE. We are also exploring the use of natural language processing tools, including large language models, to assist the process of translating existing contract wordings into InsurLE.

Prior to writing this paper, our collaboration with the insurance industry had not included collaboration with legal professionals working in the insurance industry. However, since submitting this paper, thanks to the Editors of this Journal, it is clear that there are significant benefits to be obtained by working more closely with insurance law professionals. These benefits include, but are not limited to, a more robust judicial approach to contract interpretation, a more refined attention to multiple causation of losses, and issues concerning 'facts' which are not obviously true or false. Moreover, we also have a much better appreciation that the further development and acceptance of InsurLE would be accelerated with such collaboration. We hope that such collaboration will also be of interest and benefit to the legal community as well.

---

[71] Sartor et al. (n 56) 5.

[72] T. J. Bench-Capon, 'Support for policy makers: formulating legislation with the aid of logical models' in *Proceedings of the 1st international conference on Artificial intelligence and law* (ACM 1987) online at https://dl.acm.org/doi/pdf/10.1145/41735.41756 [accessed 14 November 2024].