Logical English for Legal Applications

Robert Kowalski
Department of Computing
Imperial College London
London, UK
rak@doc.ic.ac.uk

Jacinto Dávila
Contratos Lógicos. C.A. and
Universidad de Los Andes
Mérida, Venezuela
jd@logicalcontracts.com

Miguel Calejo
logicalcontracts.com
Lisbon, Portugal
mc@logicalcontracts.com

ABSTRACT
Logical English (LE) is syntactic sugar for logic programs, which are collections of facts and rules of the form conclusion if conditions. In this paper, we focus on legal applications of LE and the use of meta- (or higher-order) predicates to represent propositional attitudes, such as permission, obligation, notification of a message and designation of the occurrence of an event. We also illustrate an integration of LE with SWISH, the online implementation of SWI Prolog.

CCS CONCEPTS
• Software and its engineering → Very high level languages; Syntax; Semantics; • Computing methodologies → Natural language processing; Knowledge representation and reasoning; • Applied computing → Law.

KEYWORDS
Logical English, logic programming, language of law

ACM Reference format:

1 Introduction
Most computer languages today are computer-oriented languages, which can be understood by humans only with special training. In contrast, Logical English (LE) [20, 21] is both a general-purpose, Turing-complete computer language, and a form of English, which can be understood by humans without any training in computing, logic or advanced mathematics.

LE is an English-like syntax for logic programs, and is inspired in part by the observation that well-written legal documents often have the form of logic programs [35, 18]. In this paper, we present the current state of LE and its integration with SWISH [39], an online implementation of SWI Prolog.

Our long-term aspiration for LE is for it to contribute to the development of more human-understandable, explainable, general-purpose computer languages of the future. But in the short term, we focus on legal applications, which potentially include expert systems, computable contracts, smart contracts on blockchains, rules as code and plain English legal writing. In this paper, we illustrate the current state of the SWISH implementation of LE, in the context of examples inspired by the

Figure 1: Clause 1.-1(1) of the British Nationality Act 1981 in LE on SWISH.

ISDA Master Agreement [21] and the loan agreement of Flood and Goodenough [12].

But first, we illustrate LE with an example from the British Nationality Act 1981 (BNA) [35]. Here is the original English expression of the very first clause of the BNA:

1.-1(1) A person born in the United Kingdom after commencement shall be a British citizen if at the time of the birth his father or mother is
(a) a British citizen; or
(b) settled in the United Kingdom.

Figure 1 shows an LE representation of 1.-1(1). The main difference between the two representations is that the conditions concerning a person’s time and place of birth are expressed as separate conditions in LE, but are embedded in the conclusion in the original English.

In the LE representation, each “template” in lines 6-12 of Figure 1 declares a fixed predicate together with its variable arguments. The predicate is a simple or compound verb interspersed with arguments that are simple noun phrases delimited by a pair of asterisks “.” The position of an argument in a template indicates its semantic role in relation to the predicate of the template.

The variable nature of an argument is signalled by an indefinite article “a” or “an” followed by a common noun or by an adjective followed by a common noun. In the current implementation, all
variables are purely mnemonic. In the future, we plan to use common nouns to indicate the types of variables, and different adjectives attached to the common noun to indicate different variables of the same type.

A template can be instantiated by replacing an argument of the template by a noun phrase or some other expression referring to an individual, including a symbol, as in a person P is a parent of a person Q if P is the mother of Q. If the template contains the keyword “that”, then the argument following the keyword can be replaced by an instance of a template. For example, the template *a person* says *that* *a sentence*, can be instantiated to Alice says that Mary says that John is the father of Alice.

The “knowledge base” named “citizenship” in lines 15-21 contains a rule representing 1.-1. The conclusion (on lines 14-15) and conditions (on lines 16-21) of the rule are instances of the templates. Indentation indicates the relative priorities of the logical connectives “and” and “or”.

A key feature of LE is its representation of variables, building on the fact that all variables in rules are universally quantified. The indefinite article “a” or “an” at the beginning of a noun phrase introduces the first occurrence of a variable in a rule, and the same noun phrase with the indefinite article replaced by the definite article “the” is used for all later occurrences of the same variable in the same rule. To reduce ambiguity, LE has no pronouns, such as “he”, “she” or “it”.

Despite its English-like appearance, the only linguistic knowledge that LE incorporates about English is:

- An expression of the form conclusion if conditions is a syntactically correct sentence if conclusion is a syntactically correct sentence and conditions is a boolean combination of syntactically correct sentences.

- Two noun phrases have the same meaning if they differ only in an initial article a, an or the.

As a matter of style, wherever possible, every English verb is associated with only a single template and is expressed in the singular and in the present tense. Every common noun is expressed in the singular and in the common case. This style avoids the need for linguistic knowledge about subject-verb agreement. The elimination of the past and future tense of verbs is made possible by eliminating the notion of current time, which constantly changes, and by replacing it with a timeline, which never changes.

In the next section, we illustrate the current state of the SWISH implementation of LE with a variant of an example from [21], which investigated the use of LE to standardise the wording of the Automatic Early Termination clauses of International Swaps and Derivatives Association (ISDA) Agreements. The example illustrates the use of meta- (or higher-order) predicates to represent permission, notification and designation of the occurrence of an event.

In the following section, we present an example from the loan agreement of Flood and Goodenough [12]. The loan agreement illustrates the use of meta-predicates to represent obligation, failure to fulfill an obligation, and curing a failure before a deadline.

2 Permission to designate an event occurrence

The Early Termination clause 6(a) of an ISDA Master Agreement allows a non-defaulting party to terminate a contract following an event of default by the other party of the contract:

If at any time an Event of Default with respect to a party (the “Defaulting Party”) has occurred and is then continuing, the other party (the “Non-defaulting Party”) may, by not more than 20 days notice to the Defaulting Party specifying the relevant Event of Default, designate a day not earlier than the day such notice is effective as an Early Termination Date in respect of all outstanding Transactions. If, however, “Automatic Early Termination” is specified in the Schedule as applying to a party, then ….

The following LE rule, which is a variant of the representation in [21] shows how the exception, signalled by the English words “If, however”, is incorporated into the representation of 6(a) as an additional condition expressing that the exception does not apply:

*it is permitted that a party designates that Early Termination in respect of all outstanding Transactions occurs at a time T3 when an Event of Default occurs with respect to an other party at a time T1 and the Event is continuing at a time T2 and the party gives notice to the other party at T2 that the Event occurs at T1 and T2 is on or before T3 and T3 and T2 are at most 20 days apart and it is not the case that the Schedule specifies that Automatic Early Termination applies to the other party for the Event of Default.*

The rule illustrates the use of that to embed one sentence inside another. This embedding, which is called meta-programming in logic programming (LP), is similar to the use of higher-order functions in functional programming and to the use of modal operators in modal logic. François Bry [6] discusses several semantics for meta-programming in LP, and proposes a novel semantics that is a conservative extension of first-order logic.

The predicates used in the rule are declared by the templates:

*it is permitted that *an eventuality*,
*a party* designates that *an eventuality*,
*an event* occurs at *a time*.
*an event* of Default occurs with respect to *a party* at *a time*.
*an event* is continuing at *a time*.
*a party* gives notice to *a party* at *a time* that *a message*.
*a date* is on or before *another date*.
*a time* and *a time* are at most *a number* days apart, the Schedule specifies that *a specification*.

Automatic Early Termination applies to *a party* for *an event* of Default.

Here *an eventuality*, *a message* and *a specification* are meta-variables, which can be instantiated by an instance of any template. With the aid of the templates, the parser expands the rule into the following Prolog clause (where a variable is represented by an underscore _ followed by a number):
It is permitted that (designates at that (229076, 229078, occurs at (Early Termination in respect of all outstanding Transactions, 229086))):

- if it is continuing at (229114, 229116, 229118),
  - of Default occurs with respect to at (229114, 229116, 229118),
- gives notice to at that (229076, 229116, 229078, occurs at (229114, 229118)) is on or before (229078, 229086),
- is not more than days after (229086, 20, 229078),
- the Schedule specifies that (Automatic Early Termination applies to for of Default (229116, 229114)).

Figure 2 shows the use of the rule to answer query one with scenario one.

The expression it is permitted that in this example resembles the permission operator of deontic logic. But in this context, it is better understood as expressing that an agent is empowered to bring about a state of affairs. In the context of 6(a), the logic of empowerment can be expressed as a rule:

an event occurs at a time T2
if it is permitted that a party designates at a time T1
  that the event occurs at T2
  and the party designates at T1 that the event occurs at T2.

Notice that there is no restriction here on the relationship between the times T1 and T2. In particular, T1 can be later than T2 in the case of a retroactive event. For example, the annulment at T2 of a marriage at T1 retroactively terminates the marriage at T1. In the case of 6(a), the Early Termination Date T3 is not retroactive.

3 Curing a failure to fulfill an obligation

The loan agreement of Flood and Goodenough (FG) exemplifies many of the characteristics of legal contracts, including the treatment of obligations, violations and remedies. Here is the first part of Clause 5, subclause(a) and the first part of Clause 6:

5. Events of Default: The Borrower will be in default under this agreement upon the occurrence of any of the following events or conditions, provided they shall remain uncured within a period of two days after notice is given to Borrower by Lender of their occurrence (such an uncured event an “Event of Default”):

(a) Borrower shall fail to make timely payment of any amount due to Lender hereunder; ...(b),...(c),...(d)....

A default will be cured by the Borrower (i) remedying the potential event of default and (ii) giving effective notice of such remedy to the Lender....

6. Acceleration on Default: Upon the occurrence of an Event of Default all outstanding payments under this agreement will become immediately due and payable....

The wording of the agreement suggests that an Event of Default occurs when one of the events (a)-(d) occurs. But this would mean that the Event of Default is retroactive, and that the acceleration of all outstanding payments in Clause 6 would have to take place in the past, with the borrower foreseeing the future.

Clearly, for Clause 6 to make sense, the Event of Default must occur at the end of the two day period for curing a potential default, in which case the original event of kind (a)-(d) is better understood as failure to fulfill an obligation. Here is 5(a) in LE:

the borrower fails on a date to fulfill an obligation if the obligation is
  that the borrower pays an amount to the lender on the date
  and it is not the case that
  the borrower pays the amount to the lender on the date.

the borrower defaults on a date D3
if the borrower must fulfill an obligation
  and the borrower fails on a date D0 to fulfill the obligation
  and the lender gives notice to the borrower on a date D1
  that the borrower fails on D0 to fulfill the obligation
  and D3 is 2 days after D1
  and it is not the case that
  the borrower cures on a date D2 the failure of the obligation
  and D2 is on or before D3.

the borrower cures on a date D the failure of an obligation if the obligation is
  that the borrower pays an amount to the lender on a date D0
  and the borrower pays the amount to the lender on a date D1
  and the borrower gives notice to the lender on a date D2
  that the borrower pays the amount to the lender on D1
  and D is the latest of D1 and D2.

Suppose the lender lends the borrower $1000, and the borrower must repay the lender in two yearly instalments:

the borrower must fulfill obligation1.

obligation1 is that the borrower pays 550 to the lender on
201 5-06-01.
the borrower must fulfil obligation2.

obligation2 is that the borrower pays 525 to the lender on 2016-06-01.

Figure 3 shows the execution of a scenario in which the borrower fails to make the first and second payments on time, but the lender notices only the second failure. The borrower attempts to cure the second failure by making the second payment, but gives the lender notice one day late.

4 Relationships with other work

LE is a controlled natural language (CNL), which is similar in spirit to ACE [10] and PENG [34], which are also implemented in Prolog. But, different from ACE and PENG, which are syntactic sugar for first-order logic, LE is syntactic sugar for a variant of pure Prolog, which is a non-monotonic, meta (or higher-order) logic. The relationship of LE to this variant of Prolog is similar to the relationship of PENG to the LP language ASP.

Compared with most other CNL languages, the main distinguishing feature of LE is its use of templates and its lack of an English dictionary and grammar. As a result, LE is closer to the computer-executable LP language into which it is translated, whereas the linguistically-based approaches are closer to ordinary natural language. Arguably, the LE approach reduces ambiguity and misunderstanding. The challenge is to ensure that this simplified LE approach is acceptable to users in practice; and if it is not, to extend LE with additional linguistic knowledge in small increments until it reaches an adequate level of acceptability.

LE can also be regarded as a domain-specific language (DSL) for legal applications. In this regard, it resembles such domain-specific languages as Blawx [27] and Oracle Policy Modelling [24], which are also rooted in LP.

In addition, LE has features in common with CNLs that do not have LP foundations. For example, RegelSpraaK [8], which has been used to automate tax law within the Dutch Tax Administration over the last decade, was initially based upon the RuleSpeak approach [40], which deliberately avoided the use of if ... then... syntax. However, over the years, The language has evolved so that all rules have the LP-like form “result if conditions”. Whereas in RegelSpraaK the results and conditions are connected using carefully composed Dutch phrases to maximize the resemblance to a natural sentence”, in LE the template declarations and syntactic constraints are used to maximise the resemblance to natural English.

Similarly, LE also resembles the natural language (NL) syntax [2] for query-answering in Cyc, a massive knowledge base of general common-sense rules and assertions. Like LE, the Cyc NL uses English-like templates to represent predicates, and it uses universally quantified rules to represent knowledge. Like LE, the Cyc NL also uses determiners to introduce variables. But because rules are written in the form if conditions then conclusion, it uses the determiner some to introduce a variable and the determiners the and that for later occurrences of the same variable in the same rule. Because the Cyc knowledge base is already written in a formal logical language, the main use of the Cyc NL is to represent queries and explanations for answers derived by means of the Cyc inference engine. In contrast, the main intended use of LE is to represent logic programs.

LE inherits its semantics [19] from the logic of LP. This is a non-monotonic logic with default negation and meta-level reasoning [5, 6]. In contrast with modal logic [28] for representing such deontic modalities as permission and obligation, LE uses meta-predicates to represent propositional attitudes more generally.

5 Discussion

Our goals for LE are that it be:

- a machine-oriented computer language,
- a human-oriented logic, and
- a controlled but natural form of English.

We have addressed the first goal by translating LE into LP, employing implicit quantifiers and infix predicates as syntactic sugar. We have addressed the second goal by building upon the logical semantics of LP with meta-predicates. The third goal is potentially the most challenging. So far, it has been addressed primarily by the use of templates, which have proved to be natural and sufficient for several real legal documents.

LE is still under development. The current implementation includes many, but not all of the features envisioned in [21]. The most important missing feature is the treatment of common nouns as types. This absence of types reflects the untyped nature of most logic programming languages including Prolog.

As already mentioned, LE employs virtually no linguistic knowledge about English grammar and vocabulary. The incorporation of even a small amount of such knowledge might go a long way towards making LE more readable.
In addition to such features of the language itself, the current implementation also lacks adequate editing, error detection and debugging tools. The current system in SWISH provides access to the Prolog debugger, which is very useful, not only for debugging, but also for explaining reasoning steps. Rendering such explanations in LE is one of the most important extensions to be introduced in the future.

Without good editing, error detecting and debugging tools, writing LE is difficult. To help with these difficulties, we plan to use the templates to guide the writer by means of a predictive editor. But, even with the help of such tools, we anticipate that writing readable LE will still be difficult. To help writers construct well-written, readable LE, we need a larger corpus of well-written examples, to serve as a guide to writing style.

In our experience, well-written legal examples, and some of the many legal applications written in pure Prolog [1, 3, 4, 6, 13, 14, 17, 22, 23, 25, 26, 29, 30, 31, 32, 33, 35, 36, 37, 38] are a good place to look for practical applications and inspiration for LE. In some cases, implementing legal applications in LE might even make them easier for humans to understand. Perhaps, more importantly, it might make them harder to misunderstand.

ACKNOWLEDGMENTS

Veselin Karadotchev implemented a variant of LE and part of the ISDA Master Agreement [16]. Ziyan Fu implemented a further variant of LE and of the FG loan agreement [9]. Aora and D2LT in London and LodgeIT in Perth helped to fund initial work on the SWISH implementation of LE. Many thanks also to Jason Morris, Giovanni Sartor, Rolf Schwitter, Matthew Waddington and Adam Wyner for helpful comments on an earlier version of this paper.

REFERENCES

requirements for compliance with law. Proc: 1st ACM International Health Informatics Symposium.


