Logical English Demonstration

Robert Kowalski
Department of Computing
Imperial College
London, UK
r.kowalski@imperial.ac.uk

Jacinto Dávila
Universidad de Los Andes
Mérida, Venezuela
jacinto@ula.ve

Logical English (LE) is a natural language syntax for pure Prolog and other logic programming languages, such as ASP and s(CASP). Its main applications until now have been to explore the representation of a wide range of legal texts, helping to identify ambiguities, explore alternative representations of the same text, and compare the logical consequences of the alternatives. The texts include portions of loan agreements, accountancy law, Italian citizenship, EU law on criminal rights, International Swaps and Derivative contracts, and insurance contracts. The current implementation in SWI Prolog can be accessed at https://logicalenglish.logicalcontracts.com.

1 Introduction to Logical English

The basic form of LE is simply syntactic sugar for pure Prolog[2], with predicates written in infix form and declared by means of templates, as in:

```
*a borrower* defaults on *a date*
```

where asterisks delimit the argument places of the predicate. Variables are signalled by the use of one of the determiners "a", "an" or "the". An indefinite determiner, "a" or "an", introduces the first occurrence of a variable in a sentence. All later occurrences of the same variable in the same sentence are prefixed by the definite determiner "the".

LE has only minimal linguistic knowledge of English. Its knowledge of English vocabulary is restricted to the determiners; the logical connectives "if", "and", "or" and "it is not the case that"; the logical pattern "for all cases in which...it is the case that..."; and the logical keyword "that". The keyword "that" identifies the use of a meta-predicate, for representing such "propositional attitudes as prohibition, obligation, belief, desire, fear, notification, etc. Indentation, rather than parentheses, is used to indicate the relative strength of binding of the logical connectives. LE has virtually no knowledge of English grammar. In particular, it does not distinguish between singular and plural nouns and verbs, and it does not know about the relationship between the different tenses of verbs. Despite these restrictions, and because it has the same expressiveness as pure Prolog, it can be used to represent a broad range of knowledge, as shown by its application to the representation of legal texts [3, 4, 5, 6, 7]. Here is an example based on the loan agreement in [1]. The SWISH implementation of the example can be found at https://logicalenglish.logicalcontracts.com/p/new_loan_with_cure.pl.

Ordinary English: The Borrower will be in default under this agreement upon the failure of the Borrower to fulfil any obligation of this agreement, provided the failure shall remain uncured within a period of two days after notice is given to the Borrower by the Lender of the failure (such an uncured event an "Event of Default").

Submitted to:
ICLP 2023

Logical English:

```
the borrower defaults on a date D2
if the borrower has the obligation that an eventuality
and the borrower fails to satisfy the obligation that the eventuality
and the lender notifies the borrower on a date D1 that
    the borrower fails to satisfy the obligation that the eventuality
and D2 is 2 days after D1
and it is not the case that
the borrower cures on or before D2 the failure to satisfy the obligation that
    the eventuality.
```

Prolog:

```
defaults_on(the_borrower, A) :-
    has_the_obligation_that(the_borrower, B),
    fails_to_satisfy_the_obligation_that(the_borrower, B),
    notifies_on_that(the_lender, the_borrower, C,
        fails_to_satisfy_the_obligation_that(the_borrower, B)),
    is_days_after(A, 2, C),
    not cures_on_or_before_the_failure_to_satisfy_the_obligation_that(the_borrower, A, B).
```

Notice that the original English suggests that an Event of Default occurs retroactively on the date D0 of the failure to fulfil an obligation. However, elsewhere the loan agreement states that "upon the occurrence of an Event of Default all outstanding payments under this agreement will become immediately due and payable". To avoid inconsistency, the LE version represents the intended interpretation as being that the default occurs on the date D2, two days after the notification of the failure on date D1.

```
22  the borrower has the obligation that the borrower pays 550 to the lender on 2015-06-01.
23  the borrower has the obligation that the borrower pays 525 to the lender on 2016-06-01.
24
25  the borrower defaults on a date D2
26      if the borrower has the obligation that an eventuality
27      and the borrower fails to satisfy the obligation that the eventuality
28      and the lender notifies the borrower on a date D1 that
29          the borrower fails to satisfy the obligation that the eventuality
30      and D2 is 2 days after D1
31      and it is not the case that
32          the borrower cures on or before D2 the failure to satisfy the obligation that the eventuality.
33
34  the borrower cures on or before a date D2  the failure to satisfy the obligation that
35          the borrower pays an amount to the lender on a date
36      if the borrower pays the amount to the lender on a date D0
37      and the borrower notifies the lender on a date D1 that
38          the borrower pays the amount to the lender on D0
39      and D0 is on or before D2
40      and D1 is on or before D2.
41
42  the borrower fails to satisfy the obligation that
43      the borrower pays an amount to the lender on a date
44      if it is not the case that
45          the borrower pays the amount to the lender on the date.
46
47  scenario payment is:
48      the lender notifies the borrower on 2016-06-04 that the borrower fails to satisfy the obligation
49          that the borrower pays 525 to the lender on 2016-06-01.
50      the borrower pays 525 to the lender on 2016-06-05.
51      the borrower notifies the lender on 2016-06-06 that the borrower pays 525 to the lender on 2016-06-06.
52
53  query defaults is:
54  which person defaults on which day.
```

Figure 1: Rules and payment scenario

Figure 1 illustrates a scenario, called "payment", in which the borrower fails to satisfy the obligations, on lines 22 and 23, to pay the lender 550 on 2015-06-01 and 525 on 2016-06-01. The lender does not notice the first failure, but notices the second failure, and gives notice to the borrower of the second failure on 2016-06-04. The borrower attempts to cure the failure, by paying the correct amount 525 and by notifying the lender of the payment within the two day period of grace. But unfortunately, the borrower notifies the lender incorrectly that the payment was made on the date of notification rather than on the date of payment.

```
Answer: the borrower defaults on 2016-6-6T0:0:0.0
true

?- answer defaults with payment.
```

Figure 2: Querying LE

Figure 2 illustrates the result of answering the combination of the stored query, called "defaults" with the scenario. An LE document can contain several stored scenarios and several stored queries, which can be combined in the SWISH query pane.

The SWISH implementation also generates explanations in response to commands of the form $answer(defaults, with(payment), le(E), R)$ as shown in figure 3. For VSC users there exist extensions for syntax highlighting and remote execution on a SWISH server. It is also possible to call the LE parser without the SWISH environment, as a standalone Prolog application. All the sources and further information are available at `https://github.com/LogicalContracts/LogicalEnglish/`.

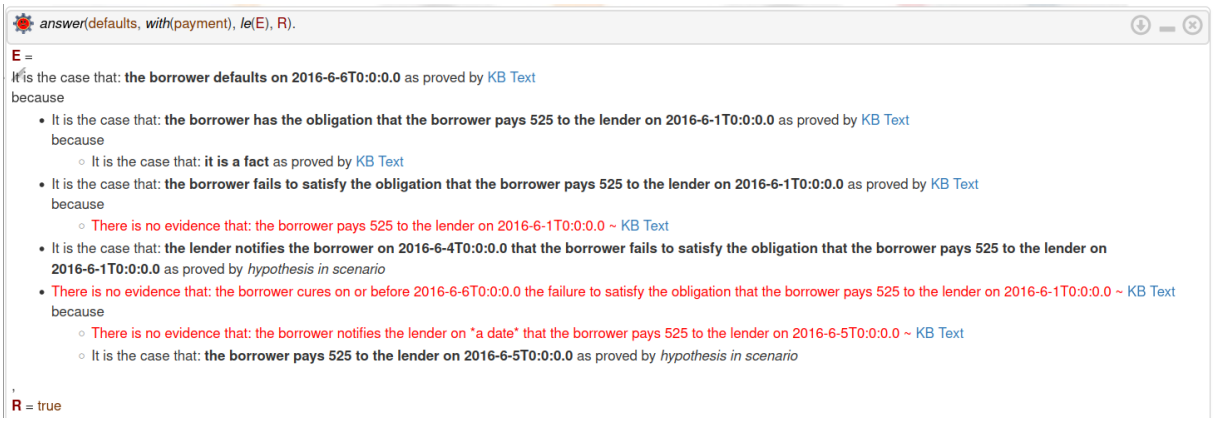

Figure 3: Explanations in LE

2 Acknowledgements

Many thanks to Miguel Calejo, Galileo Sartor, Andrew Noble, John Cummins, Fariba Sadri and Nilokai Merritt for their support and contributions to this work.

References

[1] Flood, M.D. and Goodenough, O.R., 2017. Contract as automaton: The computational representation of financial agreements. Office of Financial Research Working Paper, (15-04).

[2] Kowalski, R.: Logical English, In Proceedings of Logic and Practice of Programming (LPOP) (2020).

[3] Kowalski, R., Datoo, A.: Logical English meets legal English for swaps and derivatives. Artificial Intelligence and Law, 30(2), 163-197 (2022).

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

[5] Kowalski, R., Dávila, J., Sartor, G., Calejo, M.: Logical English for Law. In Proceedings of the Workshop on Methodologies for Translating Legal Norms into Formal Representations (LN2FR), JURIX, (2022).

[6] Kowalski, R., Sadri, F., Calejo, M., Dávila, J.: Combining Logic Programming and Imperative Programming in LPS. In: Warren, D., Dahl, V., Eiter, T., Hermenegildo, M., Kowalski, R., Rossi, F. (eds.) Prolog - The Next 50 Years. LNCS, vol. 13900. Springer, Heidelberg (2023).

[7] Sartor, G., Dávila, J., Billi, M., Contissa, G., Pisano, G., Kowalski, R.: Integration of Logical English and s(CASP), 2nd Workshop on Goal-directed Execution of Answer Set Programs (GDE'22) (2022).