(Re)Integration of Logical English and s(CASP)

Galileo Sartor[1], Jacinto Dávila[2], Alessia Fidelangeli[3] and Giuseppe Pisano[3]

[1]*University of Turin, Torino, Italy*
[2]*Universidad de Los Andes, Mérida, Venezuela*
[3]*University of Bologna, Bologna, Italy*

Abstract
This paper describes the continuing use of Logical English as a logic programming language that can be interpreted by the s(CASP) reasoner. It builds upon the previous work, and proceeds to add the possibility of expressing global constraints in a form of English that can be easily understood by users, even without any specific technical training. In particular we will integrate constraints in a legal knowledge base, and demonstrate its use. We will then tackle the possibility of integrating the Event Calculus in LE, and querying the system with s(CASP). In conclusion, we will discuss the ongoing work in integrating LE and s(CASP), and assess the process, with it's results and difficulties.

Keywords
Logic Programming, Prolog, Controlled Natural Language, Legal Rule Modelling, Explainable AI, Logical English, s(CASP)

1. Introduction

The goal of this paper is to explore the use of Logical English (LE) and its integration with s(CASP) [1] on the SWISH[1] platform, and to assess their use both for drafting legal norms in logical form and for making the system more accessible for users who lack a technical background in computing or symbolic logic. Logical English (LE)[2, 3, 4] is a general-purpose computer language[2], designed to be easily understandable by a reader of English without any technical training. Programs are expressed in LE in the form of facts and rules of the logical form *conclusion if conditions*, and they are executed by translating them into a logic programming language such Prolog or s(CASP). In this paper we continue the work on the integration of LE and s(CASP) started in [5], with a special focus on *constraints* and *event calculus*.

In the past, constraints have served the dual purpose of enriching the range of possibilities and improving the efficiency of logic programming, making it possible to reduce the space of solutions of a logic program with a very concise syntax. Indeed, the main contribution provided by the s(CASP) project is the efficient integration of constraints in a SLD resolution-

3rd Workshop on Goal-directed Execution of Answer Set Programs (GDE'23), July 10, 2023
galileo.sartor@unito.it (G. Sartor); jd@logicalcontracts.com (J. Dávila); alessia.fidelangeli2@unibo.it (A. Fidelangeli); g.pisano@unibo.it (G. Pisano)
0000-0001-6355-851X (G. Sartor); 0000-0003-1053-8763 (J. Dávila); 0000-0002-8281-4176 (A. Fidelangeli); 0000-0003-0230-8212 (G. Pisano)

[1]https://swish.swi-prolog.org/
[2]https://github.com/LogicalContracts/LogicalEnglish/

like procedure for ASP semantics. However, our original work in [5] did not provide a way to express global constraints in LE, and thus in s(CASP).

In ASP global constraints can be expressed with the syntax:

$$:-\mathrm{p}, \mathrm{q}$$

with the intended aim to discard all the models containing both the literals p and q. LE has been extended to also allow the encoding of these constraints in natural language with the construct:

it must not be true that p and q

where p and q represent a template in the language. For example, it is possible to write sentences as intuitive and easy to understand as *it must not be true that a person is infallible and the person is capable of making mistakes*. We will examine in Section 2 the potential applications within the legal field.

Event Calculus (EC) [6] is a very well known formalism to model and reasoning about events and their logical causes and consequences. A theory of change is normally made from the combination of the basic axioms of EC with rules that describe how events initiate and terminate fluents, i.e. properties of a system that may change through time. Many experiments have been performed in the LP and ASP communities around EC. In LE, we are aiming at supporting the description of those rules, leaving the axioms of the EC to work implicitly behind the scene. The following listing 1 could be a first approximation to those supporting axioms in the syntax of Prolog/ASP:

```
holds(Fluent, T) :-
    happens(Event, T1),
    T1 #< T,
    initiates(Event, Fluent, T1),
    not is_interrupted_between(T1, Fluent, T).

is_interrupted_between(T1, Fluent, T2) :-
    happens(Event, T3),
    terminates(Event, F, T3),
    T1 #< T3,
    T3 #=< T2.
```

Listing 1: Event Calculus Basic Axioms in s(CASP)

Notice the time variables “constrained” by the use of the #<, #=< operators in s(CASP). This is a particularly powerful device that ASP brings to the users of EC, as it is now possible to reason about partially or totally undefined times.

In the following sections we will analyse two examples written in LE and interpreted by s(CASP). In Section 2 we explore LE’s utility for the legal domain and the added features given by s(CASP) integration. In Section 3, we show how LE can already be used to describe the rules of a particular domain that could be combined, transparently to users, with those basic axioms of EC to reason about events, fluents and their timing. Section 4 concludes the work.

2. Global constraints: The NILDE Project

Logical English can be used to model legal norms in logical form. As argued, for example, by [7], some of the needs for legal modelling are (1) isomorphism, understood as a correspondence between the legal source and the knowledge base, preserving possible interpretations, references and connections, and (2) defeasibility, which enables reasoning with rules and exceptions, where the effect of a legal rule may be blocked by the applicability of another rule.

With respect to isomorphism, there may be conditions in the knowledge base that appear in multiple different rules. This happens because the rules are independent from one another, so there is a certain attention one needs to have in modelling the right sequence of statements. This is mainly an issue in domains such as the legal one, where the choice is often to try and maximise isomorphism. However, the application of legal provisions frequently depends on general conditions, that apply to a whole document, or to a larger section of the document.

We will now see how this issue could be addressed by introducing global constraints, and how this can help a legal expert in writing a knowledge base in Logical English.

We used Logical English and s(CASP) to model legal rules in the field of tax law in the NILDE project. In particular the choice was made to model sections of the conventions against double taxation between Italy and France.

The addition of features from s(CASP) in this knowledge base presents a number of advantages, such as:

- having incomplete (or unknown) information, and leveraging the abducibles to present the possible solutions;
- adding constraints in the knowledge base to improve efficiency and readability;
- leveraging the s(CASP) explanation module to show the Answer Set and it's natural language explanation.

All this remains possible while also relying on the natural language structure of Logical English to make the knowledge base itself easier to read and write.

Here we will focus on the use of global constraints, in particular to set rules that are defined globally. This simplifies the rules themselves, as it is no longer necessary to carry these conditions in multiple different rules, making it easier for the legal expert in charge of writing the rules in a way that is more natural.

2.1. The NILDE Project

The objective of the NILDE project is to create a computable model of the French-Italian convention against double taxation in Logical English. Conventions against double taxation are bilateral tax treaties which are binding only for the states which signed them. The objective of these conventions is to decide on the allocation of taxing rights between the two states: they specify the state which can tax a certain income, the withholding taxes that can be applied, and other measures affecting taxation in cross-border situations. Conventions against double taxation cover taxes on income and taxes on capital, as these taxes are usually personal taxes and rely on worldwide taxation for residents, thus potentially generating double taxation. Through the attribution of taxing rights to one of the two states double taxation is avoided. The

French-Italian convention entered into force on 1 May 1992 and it covers several Italian and French taxes on income or capital.

NILDE's knowledge base is in development, and will enable the user to understand which state is entitled to tax a certain income. This assessment is done on the basis of a number of facts that the user has to provide as input (as we will see in the example code, represented as the scenario). These facts are, for example: where the taxpayer is resident, what is the source of the income, in which country is that source located, etc. The main goal of the questions is to provide the system with the necessary knowledge to tell the user which state is entitled to tax a certain income (e.g., the income received by an Italian resident in relation to the rental of an immovable property located in France is taxable in France). The solution will then be provided to the user both concisely and with the identification of the intermediate steps and facts that led to the solution.

We decided to model the area of conventions against double taxation in Logical English as these legal sources provide for a closed and self-standing regulatory system, with limited references to national sources. So, modeling a limited number of rules, it is possible to have a system which provides answers to taxpayers acting in transnational situations. Moreover, such conventions have a clear structure and this simplifies the modeling phase. In addition, the text of most conventions is unlikely to be modified or updated, which reduces the risk of changes in the system that would require modifications of the modeled rules. Finally, most of the conventions drafted between western countries are concluded on the basis of the OECD Model Convention. This is the basis for negotiation and application of bilateral tax treaties between OECD countries. Hence, the work done with NILDE may be used to model other conventions based on the same OECD Model.

2.2. LE and s(CASP) example

In the NILDE knowledge base there are many different rules that may determine where the interested party may be taxed for a specific income. As mentioned in Section 2.1, a legal expert was given the task of formalising the convention against double taxation between Italy and France.

The convention applies only if a person is resident in one of the two states (i.e., the "state of residence"), while producing or receiving income in or from the other (i.e., the "state of source"). In fact, if the income is produced in the state where the person is resident there is no need to rely on a convention to decide whether the state of source or the state of residence has the right to tax that income. This condition is not made explicit in the rules of the convention but and is assumed as a given.

The issue then rises: how to model this in a way that is efficient and clear to read and understand?

An initial solution was to add *entry point* rules, that had this condition, and then called the other rules. While this works, there might be more to gain from having global constraints, so that you know that the constraint is applied to every query.

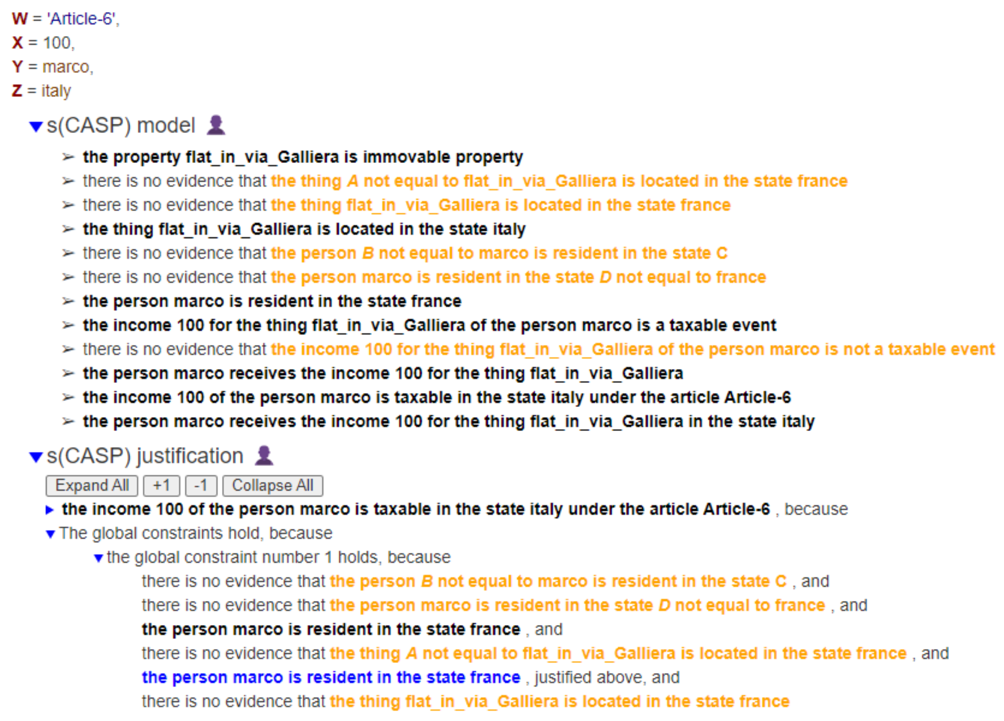
What we ended up with was a (small) set of rules that defined which state the income was originating from (i.e. state of source), then added a global constraint that negated the possibility for the subject to be resident and receiving the income from the same state.

```
 1 the target language is: scasp.
 2
 3 the templates are:
 4 *an income* of *a person* is taxable in *a state* under *an article*.
 5 *a person* receives *an income* for *a thing* in *a state*.
 6 *a person* receives *an income* for *a thing*.
 7 *a thing* is located in *a state*.
 8 *a person* is resident in *a state*.
 9 *a property* is immovable property.
10 *an income* for *a thing* of *a person* is a taxable event.
11 *an income* for *a thing* of *a person* is not a taxable event.
12
13 the knowledge base double_taxation includes:
14 an income of a person is taxable in a state S under Article 6
15     if the person receives the income for a property in the state S
16     and the property is immovable property.
17
18 a person receives an income for a property in a state S
19     if the income for the property of the person is a taxable event
20     and the property is located in the state S.
21
22 an income for a thing of a person is not a taxable event
23     if the person receives the income for the thing
24     and it is not the case that
25       the income for the thing of the person is a taxable event.
26
27 an income for a thing of a person is a taxable event
28     if the person receives the income for the thing
29     and it is not the case that
30       the income for the thing of the person is not a taxable event.
31
32 it must not be true that
33 a person is resident in a state
34 and a property is located in the state
35 and an income for the property of the person is a taxable event.
36
37 scenario one is:
38 marco is resident in france.
39 marco receives 100 for a flat in via Galliera 3.
40 flat in via Galliera 3 is located in italy.
41 flat in via Galliera 3 is immovable property.
42
43 query one is:
44 which income of which person is taxable in which state under which article.
```

Listing 2: Sample from the NILDE knowledge base

In Listing 2 the rule in line 32, *it must not be true that ...*, defines the global constraint. In this example the goal is to determine that the State in which the person is resident, and the one where a particular income is produced are not the same. This condition would have previously been included in the specific rule, but since the knowledge base is fairly large it is more convenient to write it separately as a constraint.

In the generated s(CASP) program the global constraint is translated as:


W = 'Article-6',
X = 100,
Y = marco,
Z = italy

▼s(CASP) model

- the property flat_in_via_Galliera is immovable property
- there is no evidence that the thing *A* not equal to flat_in_via_Galliera is located in the state france
- there is no evidence that the thing flat_in_via_Galliera is located in the state france
- the thing flat_in_via_Galliera is located in the state italy
- there is no evidence that the person *B* not equal to marco is resident in the state C
- there is no evidence that the person marco is resident in the state *D* not equal to france
- the person marco is resident in the state france
- the income 100 for the thing flat_in_via_Galliera of the person marco is a taxable event
- there is no evidence that the income 100 for the thing flat_in_via_Galliera of the person marco is not a taxable event
- the person marco receives the income 100 for the thing flat_in_via_Galliera
- the income 100 of the person marco is taxable in the state italy under the article Article-6
- the person marco receives the income 100 for the thing flat_in_via_Galliera in the state italy

▼s(CASP) justification

Expand All | +1 | -1 | Collapse All

▶ the income 100 of the person marco is taxable in the state italy under the article Article-6 , because

▼The global constraints hold, because

▼the global constraint number 1 holds, because

there is no evidence that the person *B* not equal to marco is resident in the state C , and
there is no evidence that the person marco is resident in the state *D* not equal to france , and
the person marco is resident in the state france , and
there is no evidence that the thing *A* not equal to flat_in_via_Galliera is located in the state france , and
the person marco is resident in the state france , justified above, and
there is no evidence that the thing flat_in_via_Galliera is located in the state france

Figure 1: NILDE example: s(CASP) justification for query one in Listing 2

```
false :-
    is_resident_in(A, B),
    is_located_in(C, B),
    for_of_is_a_taxable_event(_, C, A).
```

In this way the constraints that apply to larger portions of the knowledge base can be expressed separately, thus being more isomorphic to the original source document.

Figure 1 shows s(CASP justification) for the query at line 33 in Listing 2. Marco's income is taxable in Italy as per Article 6 of the Convention since the source of income is in Italy and he is resident in France.

3. Event Calculus: The Fox and the Crow

The ancient fable of the fox and the crow, written around the sixth century B.C. and attributed to Aesop, is a fantastic pedagogical resource. In [8] (chapter 3), Kowalski uses it to illustrate how to formalise the beliefs of a proactive agent and how she would reason with those beliefs to deduce plans to achieve her goals.

Here we state the beliefs of the fox in LE, but restricted only to her knowledge of how to get the crow to release a cheese he holds and how she, the fox, could have it. The rules in listing 3 describe the relevant beliefs in LE.

```
the target language is: scasp.

the event predicates are:
    *an animal* praises *an animal*.
    *an animal* sings.
    *an animal* picks up *an object*.
    it all begins.

the fluents are:
    *an animal* has *an object*.
    *an animal* is near *an object*.

the knowledge base foxec includes:

it becomes the case that
    an animal has an object
when
    the animal picks up the object
if the animal is near the object.

it becomes the case that
    the fox is near the cheese
when
    the crow sings.

it becomes the case that
    the crow is near the cheese
when
    it all begins.

it becomes not the case that
    the crow has the cheese
when
    the crow sings.

the crow sings, at a time T
    if the fox praises the crow, at T.

it all begins, at 1.
the crow picks up the cheese, at 2.

scenario one is:
    the fox praises the crow, at 4.
    the fox picks up the cheese, at 5.

query one is:
    which animal has which object, at which time.
```

Listing 3: The Beliefs of the Fox (in LE)

These rules are automatically mapped into the s(CASP) syntax, as shown in listing 4 (with some minor editing of the variables names in the query):

```
:-module('foxec-scasp', []).
:- set_prolog_flag(scasp_lang, en).
% s(CASP) Programming
:- use_module(library(scasp)).
% Uncomment to suppress warnings
:- style_check(-discontiguous).
:- style_check(-singleton).
:- set_prolog_flag(scasp_forall, prev).
:- dynamic picks_up/2, is_near/2, it_all_begins/0, praises/2, has/2, sings/1.
#pred picks_up(B,C) :: ' @(B:animal)  picks  up  @(C:object) '.
#pred is_near(B,C) :: ' @(B:animal)  is  near  @(C:object) '.
#pred it_all_begins :: ' it  all  begins '.
#pred praises(B,C) :: ' @(B:animal)  praises  @(C:animal) '.
#pred has(B,C) :: ' @(B:animal)  has  @(C:object) '.
#pred sings(B) :: ' @(B:animal)  sings '.
initiates(picks_up(A, B), has(A, B), C) :-
    holds(is_near(A, B), C).
initiates(sings(the_crow), is_near(the_fox, the_cheese), _).
initiates(it_all_begins, is_near(the_crow, the_cheese), _).
terminates(sings(the_crow), has(the_crow, the_cheese), _).
happens(sings(the_crow), A) :-
    happens(praises(the_fox, the_crow), A).
happens(it_all_begins, 1).
happens(picks_up(the_crow, the_cheese), 2).
/* Scenario one */
happens(praises(the_fox, the_crow), 4).
happens(picks_up(the_fox, the_cheese), 5).
% */
/** <examples>
?- ? holds(has(Who,What),When).
**/
```

Listing 4: The Fox's beliefs in SCASP notation

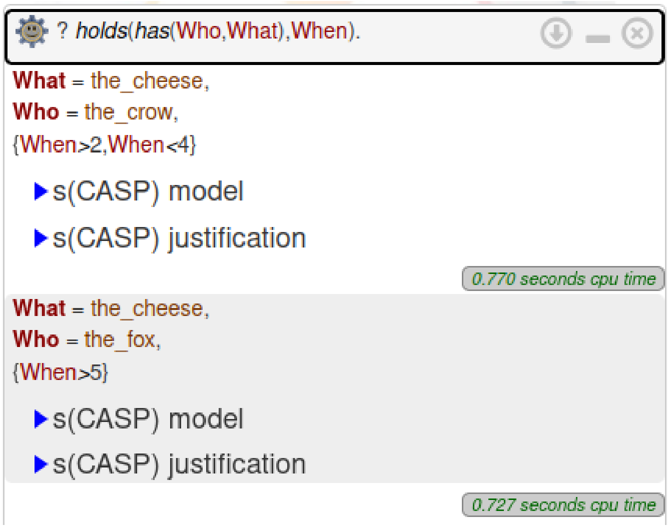
By combining these sources[3] with the basic axioms of EC (shown in the previous section), s(CASP) can answer that very general query about who has what object when, as shown in Fig 3, a query which a regular Prolog engine cannot answer:

4. Opportunities for a better integration LE-s(CASP)-SWISH

This paper extends the work in [5] on the use of Logical English as a logic programming language that can be interpreted by the s(CASP) reasoner. In particular, we added the possibility of expressing global constraints in a form of English understandable by users without any specific technical training, and showcased its potentialities in the legal domain. We also tackle the possibility of integrating Event Calculus in LE, giving users the possibility to straightforwardly reason about events, fluents and their timing.

The points for improvement are still many and may require some technical agreements between the s(CASP), SWISH and LE teams. There is an ongoing discussion about how to

[3] https://swish.swi-prolog.org/p/foxec-scasp-extended.pl



```
? holds(has(Who,What),When).
What = the_cheese,
Who = the_crow,
{When>2,When<4}
▶s(CASP) model
▶s(CASP) justification
                                        0.770 seconds cpu time
What = the_cheese,
Who = the_fox,
{When>5}
▶s(CASP) model
▶s(CASP) justification
                                        0.727 seconds cpu time
```

Figure 3: Querying who has what when

use the html and human outputs that SWISH produces for s(CASP), also in LE[4], as a first step towards the common handling of multiple natural languages. There is also another discussion about provisions for secure access to a typical SWISH installation to support the use of libraries and components in SWISH that must normally be left out of the sandbox[5]. Other issues, such as discrepancies in syntax between s(CASP) standalone and SWISH-embedded s(CASP) must also be addressed.

Acknowledgments

The authors wish to thank Prof. Bob Kowalski, for his active leadership of the Logical English project, and for his encouragement to explore its applications to many different domains.

The work has been supported by the "CompuLaw" project, funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant Agreement No. 833647), and the "NILDE" project, funded by the University of Bologna under the AlmaIdea 2022 programme.

References

[1] J. Arias, M. Carro, E. Salazar, K. Marple, G. Gupta, Constraint answer set programming without grounding, Theory and Practice of Logic Programming 18 (2018) 337–354. doi:10.1017/S1471068418000285.

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

[4] https://swi-prolog.discourse.group/t/reusing-html-and-human-output-from-scasp/6158/1

[5] https://swi-prolog.discourse.group/t/pengines-js-pengine-create-call-certain-options-are-ignored/714/8

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

[4] R. Kowalski, J. Dávila, C. L. CA, M. Calejo, Logical english for legal applications, Conference: XAIF, Virtual Workshop on XAI in Finance (2021).

[5] G. Sartor, J. Davila, M. Billi, G. Contissa, G. Pisano, R. Kowalski, Integration of logical english and s(casp), in: J. Arias, R. Calegari, L. Dickens, W. Faber, J. Fandinno, G. Gupta, M. Hecher, D. Inclezan, E. LeBlanc, M. Morak, E. Salazar, J. Zangari (Eds.), Proceedings of the International Conference on Logic Programming 2022 Workshops co-located with the 38th International Conference on Logic Programming (ICLP 2022), Haifa, Israel, July 31st - August 1st, 2022, volume 3193 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022. URL: https://ceur-ws.org/Vol-3193/paper5GDE.pdf.

[6] R. A. Kowalski, M. J. Sergot, A logic-based calculus of events, New Generation Computing 4 (1989) 67–95. doi:10.1007/BF03037383.

[7] T. F. Gordon, G. Governatori, A. Rotolo, Rules and norms: Requirements for rule interchange languages in the legal domain, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, pp. 282–296. doi:10.1007/978-3-642-04985-9_26.

[8] R. Kowalski, Computational Logic and Human Thinking: How to be artificially Intelligent, Cambridge University Press, London, 2011.