ADVISORY AND TRAINING SYSTEM FOR INTERNET-RELATED PRIVATE INTERNATIONAL LAW

D3.4 – InterLex rule base on EU legislation (beta version)

Grant agreement number	800839
Project name/acronym	InterLex
Project title	Advisory and Training System for Internet-related Private International Law
Website	www.interlexproject.eu
Contractual delivery date	31/08/2019
Actual delivery date	04/10/2019
Contributing WP	3
Dissemination level	Public
Deliverable leader	UNIBO
Contributors	UNIBO, UNITO, APIS

This project has received funding from the European Union's Justice Programme (2014-2020) under Grant Agreement Number 800839

Document History

Version	Date	Author	Partner	Description
0.1	01/01/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Rule base of Brussels Regulation
0.2	01/05/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Rule base of Rome I Regulation
0.3	01/07/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Rule base of Rome II Regulation
0.4	6/09/2019	Giuseppe Contissa	UNIBO	First complete draft
0.5	13/09/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Revised internal version
0.6	27/09/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Second revised internal versions
1.0	01/10/2019	Giuseppe Contissa Francesca Lagioia Giovanni Sartor	UNIBO	Final version

Contributors

Partner	Name	Partner	Description
APIS	Hristo Konstantinov	Reviewer	All sections
UNITO	Luigi Di Caro	Reviewer	All sections

Table of Contents

List of Figures

N/A

List of Tables

N/A

List of Acronyms

PIL	Private International Law
UNIBO	UNIversity of BOlogna
WP	Work package

Executive Summary

This document constitutes Deliverable 3.4 of the InterLex project. Its primary objective is to provide a report on the development of a formal representation of the main EU regulations in the area of Private International Law, in Prolog rules, to be used as core content for the Decision Support Module (DSM) of the InterLex Platform.

Section 1 introduces the application domain of EU PIL regulations and describes the motivations behind the selection of parts of the legislation to be modelled.

Section 2 describes the structure and content of the rule base, and the methodology adopted for its development. It is divided into 3 main subsections.

Subsection 2.1 provides an overview of the Prolog representation, covering the high-level structure of the rule base for each of the three main source regulations.

Subsection 2.2 explains the methodological approach followed in the development of the rule base, including in particular the development of predicates, the management of exceptions, and the development of the meta-interpreter in order to provide explanations (the rationale for the assessment of the goal).

Subsection 2.3 presents a use case scenario and a demonstration of the functioning of the Prolog system.

Section 3 presents our conclusions, summarizing the most relevant parts of all the work done and forming the basis for the work that will be carried out in the future.

The deliverable is completed with an appendix containing (as an embedded document) the complete rule base in Prolog, including the Prolog meta-interpreter for providing explanations.

1. Introduction: the EU legislation corpus for PIL

This deliverable presents the first release of the Interlex rule base. It is a beta version, covering the main instruments of EU legislation on Private International Law (PIL).

In particular, the rule base representation covers 3 main legislative EU instruments: the Regulation (EU) No 1215/2012 on jurisdiction and the recognition and enforcement of judgments in civil and commercial matters (recast) (Brussels Regulation); the Regulation (EC) No 593/2008 on the law applicable to contractual obligations (Rome I); and the Regulation (EC) No 864/2007 on the law applicable to non-contractual obligations (Rome II).

The reference documents adopted as sources for the representation were the digital copies in PDF format and in the English language available on the EUR-Lex website. The development of the representation started with a legal analysis of the EU regulations, aimed at identifying the core part of the laws to be represented in relation to the goals of the InterLex project. The following parts were selected:

- Brussels Regulation: the core part of the regulation that has been selected for the representation includes all the sections that are relevant for the identification of jurisdiction, namely Chapter I (Scope and Definitions); Chapter II (Jurisdiction), Sections 1 – 7; and Chapter V (General Provisions), Articles 62,63 and 65.

- Rome I Regulation: the whole content of the regulation has been selected for the representation, with the exception of Article 18 and Articles 21-25.

- Rome II Regulation: the whole content of the regulation has been selected for the representation, with the exception of Articles 15-18, 21-22, 26-31.

As a general rule, the articles relevant to the assessment of jurisdiction and applicable law have been represented. The missing articles contain either clarifications or examples (e.g. non-closed definitions). The representation of these rules will be evaluated for future versions of the rule base.

Then, each norm from the core parts of the legal source texts was represented in a correspondent rule (or set of rules) by specialised knowledge engineers supported by legal experts. The legal analysis often involved the interpretation of complex legal rules, and the selection of one interpretation among several possible ones. This analysis was carried out with the support of legal commentaries ([1], [2]).

No deviations were made, as long it was possible, from the original structure of the text, even when it was redundant or convoluted (as regard to the logic representation), in order to preserve isomorphism (see InterLex Deliverable 3.1, section 1)

2. The Prolog representation

2.1. HIGH-LEVEL STRUCTURE OF THE RULE BASE

The language adopted for the representation is SWI-Prolog, in accordance with the selection carried out at the beginning of the project and documented in Deliverable 3.1. However, the similarity of SWI-Prolog implementation to the ISO-Prolog standard also means that the Prolog code is not entirely specific to the chosen Prolog implementation, and can be easily re-used with other versions of the language. The approach adopted for the modelling is described in Deliverable 3.1, section 3.4.

The Interlex Rule Base is logically organized according to the specific chapters and sections of the source legal documents that have been selected for the representation. This is mirrored in the code base by having the different sections split into separate files.

The following are the main goals and the entry levels for each of the represented sources:

- Brussels Regulation:

```
hasJurisdiction(Country, Court, ClaimId, Law)
```

- Rome Regulation:

```
applicableLaw(Article, Country, ContractId, Law)
```

- Rome II Regulation:

```
applicableLawNonContract(Article, Country, ObligationId, Law)
```

The top-level goal is immediately followed by the main exceptions and conditions, which must be checked before importing the rules contained in other chapters and sections. As an example, this is a fragment from the Brussels Regulation representation:

```
hasJurisdiction(Country, Court, ClaimId, brusselsRegulation):-

    brusselsRegulationApplies(ClaimId, brusselsRegulation),

    \+exception(hasGeneralJurisdiction, _, ClaimId),

    hasGeneralJurisdiction(Country, Court, ClaimId, brusselsRegulation).
```

In this example, the top-level goal (the assessment of the jurisdiction according to the Brussels Regulation) can be verified if: the premises (contained in Chapter 1 of the Regulation) apply, there are no exceptions to the general jurisdiction rules, and at the same time, the general jurisdiction rules apply. The same approach has been adopted in the representations of the Rome I and Rome II Regulations.

2.2. THE MODELLING APPROACH

2.2.1. The conversion process

The process of converting the initial legal text into rules involved several different stages, as required by commonly adopted Prolog-based rule-base creation processes.

This work was carried out following a sequence of iterative stages: the first stage consisted in the development of what is usually called «Pseudo-Code», that is rewriting the initial legal text in an intermediated controlled natural language (English) that makes logical structures and connectors explicit, while keeping as much as possible of the original semantics. This is the stage that must be supervised by legal experts. During this stage, specific guidelines were followed, taking into account the need for further processing of the representation. More in detail:

a) the knowledge engineer tried as much as possible to keep all the original sentences in such a way that a rule (or more rules) matches a sentence or expanded sentences;

b) the connectives (or, and, etc.) were employed by repeating the relevant part of a sentence or creating a new sentence;

c) the relatives (which, who, etc.) were developed and explicated;

d) pronouns were replaced by their referred nouns or lexical expressions;

e) irrelevant terms and repetitions were suppressed;

f) where necessary, the syntax of sentences were simplified;

g) whenever possible, use was made of equivalent terms to minimize the number of derived predicate names;

h) whenever compound words were found, the sequences of words were replaced by single terms.

In the second stage of the work, we adopted a process starting from the "pseudo-code" and up to formalization in Prolog rules.

This stage was broken into three successive steps:

Step 1 – On the basis of an initial analysis of the list of nouns, verbs and adjectives contained in the source law texts, relevant lexical terms were identified, and a basic taxonomy was developed of the concepts of PIL, their properties and relations.

Step 2 – Then, it was evaluated whether to include these concepts as Prolog predicate names or as arguments. Since a lexical term can be formalized as a different object type in the logical rules: predicate with different arities, functions, constants, etc., the taxonomy was used, and when necessary updated, with a view to maintain consistency during the translation of the whole corpus of legal texts.

Step 3 - The final formalization from pseudo-code to Prolog rules was carried out. We adopted the methodology described in D3.1, section 3.4. We also decided to introduce structural elements into the rule base, in the form of predicates, to refer to corresponding structural elements of the source legislative text (chapters and sections). Consider for example the following fragment of rules, extracted from the representation of the Brussels regulation, section 7:

```
hasJurisdiction7_1to7(Country, Court, ClaimId, brusselsRegulation):-
    hasJurisdiction7_1(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_2(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_3(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_4(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_5(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_6(Country, Court, ClaimId, brusselsRegulation);
    hasJurisdiction7_7(Country, Court, ClaimId, brusselsRegulation).

hasJurisdiction7_1(Country, Court, ClaimId, brusselsRegulation):-
    claimObject(ClaimId, contract, ContractId),
    placeOfPerformance(Country, Court).

hasJurisdiction7_2(Country, Court, ClaimId, brusselsRegulation):-
    claimObject(ClaimId, tort),
    eventOccurredOrMay(Country, Court).
```

In this example, the introduction of predicates in the form `hasJurisdiction7_1, 7_2, /_3, etc`), has a twofold purpose: 1) to maintain the isomorphism of the representation, keeping a close connection between the structure of the source texts and the structure of the representation, and 2) to keep trace of the articles in the execution process, so as to simplify debugging and validation of the rule base, and even more importantly, to improve the quality of explanations provided by the meta-interpreter (see section 2.2.3 below).

The same approach was adopted for the Rome I and Rome II Regulations, with the inclusion of an additional argument, Article, that is assigned to the article that verifies the Goal. For consistency, the article reference in the predicate name was kept in the Rome Regulations, and the Brussels representation will be updated.

In the code, comments are used mostly as references for internal development processes. Prolog can also build documentation in HTML from a subset of the documentation, so some testing was carried out to keep the original text embedded in the source code, and to be able to view the relevant code and legal text quickly.

The main testing for the codebase is based on a small number of cases, and is automated via a Python wrapper.

2.2.2. Defeasibility and exceptions

It is commonly agreed that legal reasoning is a defeasible kind of reasoning. In the EU PIL corpus, defeasibility occurs by means of expressions such as: “unless proved otherwise”; “unless otherwise agreed”; “unless …”; “except in those cases”; “with the exception of”; “subject to…”; “notwithstanding”; etc.

In Prolog, defeasibility is usually managed by using negation by failure (See Deliverable 3.1, section 3.4.3). Hence, each of the above items has to be accounted for using negation by failure.

2.2.3. Non-ISO Meta-interpreter

The main non-ISO structure in the rule base code is the meta-interpreter. A Prolog meta-interpreter is a Prolog program that takes a Prolog goal and another Prolog program, then proceeds to attempt to prove the goal against the second Prolog program, according to the rules given by the first program.

At a simple level, the meta-interpreter could be simply `prove(Goal) :- call(Goal)`, but this would not give much information, it would simply call a goal to prove it.

Below we present a revised version of the prototype meta-interpreter presented in Deliverable 3.1, section 3.4.4. We built this simple (but expandable) meta-interpreter to generate a proof tree tracing the execution of the program, by printing a log of the evaluated predicates, as a first attempt at providing a user-friendly explanation functionality.

A goal that is found to be true during the exploration of the proof tree (i.e. an asserted predicate) is a fact, and it is presented as such by the meta-interpreter.

```
% Metainterpreter
solve(true, [fact]) :- !.
solve((A,B), Result) :- !, solve(A, ARes), solve(B, BRes),
                           append(ARes, BRes, Result).
solve((A;B), Result) :- solve(A, Result); solve(B, Result).
solve(member(A,B), [member(A, B)]) :- !, call(member(A,B)).
solve(\+(A), [not(A)]) :- !, call(\+(A)).
solve((A)\=(B), [doNotUnify(A, B)]) :- !, call((A)\=(B)).
solve(A, [A|[Res]]) :- clause(A,B), solve(B, Res).
```

2.3. USE OF THE RULE BASE

In this section, we present an example showing the use of the SWI-Prolog engine in combination with the rule base based on a set of facts representing a hypothetical scenario involving the application of EU rules on Private International Law.

2.3.1. Use case scenario: Provision of services

Let us consider the following hypothetical scenario:

Silva Trade, a (natural) person located in Luxembourg, signs a contract with WoodFloor, a company located in Turin, Italy: WoodFloor agrees to provide its services to Silva, in the cities of Vienna and Paris, and Silva agrees to pay a fee for the services. However, the services are not provided as expected by Silva, and she refuses to pay the fee. WoodFloor decides to sue Silva.

The question to be assessed is: which court has jurisdiction for the claim?

In Prolog, the above facts may be represented as follows:

```
assert(claimMatter(claimId7_1, civilCommercial)).
assert(claimObject(claimId7_1, contract, contractId7_1)).
assert(contractTypeConsideration(contractId7_1, consumer)).
assert(contractType(contractId7_1, provisionOfServices)).
assert(personDomicile(silvaTrade, luxembourg, _)).
assert(personDomicile(woodFloor, italy, turin)).
assert(personNature(silvaTrade, natural)).
assert(personNature(woodFloor, legal)).
assert(personRole(silvaTrade, claimId7_1, defendant)).
assert(personRole(woodFloor, claimId7_1, claimant)).
assert(personType(silvaTrade, consumer)).
assert(placeOfProvision(austria, vienna)).
assert(placeOfProvision(france, paris)).
```

The query will be represented as follows:

```
hasJurisdiction(Country, Court, claimId7_1, brusselsRegulation).
```

The Prolog system provides three alternative solutions for the query:

```
luxembourg, _ ; france, paris ; austria, vienna
```

In order to verify the top-level goal of this chain (`hasJurisdiction(luxembourg, _1596, claimId7_1, brusselsRegulation)`), Prolog has to verify its main conditions. The meta-interpreter collects such conditions at the level just under the top goal, as part of a nested list:

```
[hasJurisdiction(luxembourg, Court, claimId7_1, brusselsRegulation),
[brusselsRegulationApplies(claimId7_1, brusselsRegulation),
[not(exception(brusselsRegulationApplies(claimId7_1, brusselsRegulation),
9548)), claimMatter(claimId7_1, civilCommercial), [fact]],
not(exception(hasGeneralJurisdiction, 9484, claimId7_1)),
hasGeneralJurisdiction(luxembourg, Court, claimId7_1, brusselsRegulation),
[personRole(silvaTrade, claimId7_1, defendant), [fact],
personDomicile(silvaTrade, luxembourg, Court), [fact],
memberState(luxembourg), [fact]]]]
```

The initial trace provided by the meta-interpreter is then formatted by the Python wrapper as follows:

```
TOP GOAL: luxembourg, 1
-> hasJurisdiction(luxembourg, _1596, claimId7_1, brusselsRegulation)
1)  -> brusselsRegulationApplies(claimId7_1, brusselsRegulation)
        -> not(exception(brusselsRegulationApplies(claimId7_1,
                                  brusselsRegulation), _1710))
        -> claimMatter(claimId7_1, civilCommercial)
            -> fact
2)  -> not(exception(hasGeneralJurisdiction, _1646, claimId7_1))
3)  -> hasGeneralJurisdiction(luxembourg, _1596, claimId7_1,
                              brusselsRegulation)
        -> personRole(silvaTrade, claimId7_1, defendant)
            -> fact
        -> personDomicile(silvaTrade, luxembourg, _1596)
            -> fact
        -> memberState(luxembourg)
            -> fact
```

In this explanation, the Prolog meta-interpreter correctly verifies the main goal, with the value country= Luxembourg, and court= unknown (_). The system then displays the predicates it verified to reach that goal (identified as 1), 2), and 3) in the code lines above), indented to represent their logical chaining. The same happens to every intermediate goal that may need to be verified (in this case, the sub-goals are those identified as 1), 2), and 3), and their immediate sub-sub-goals).

The bottom-level statements that are verified by the Prolog engine correspond to the facts asserted at the beginning of the example.

3. Conclusions

The complexity of source norms has sometimes limited the isomorphism of logical representation, in particular for the management of norms constituting implicit exceptions of general norms. However, the issue has been solved with the solution proposed in Deliverable D3.1 and applied in section 2.2.2 of the present Deliverable.

Besides, we were aware that the domain of PIL would be particularly difficult to formalize, especially when compared to those domains that are the traditional fields of the application of knowledge-based systems in law, fields such as tax legislation, administrative regulations, provision of benefits, etc.

On the one hand, tax legislation is commonly considered "complex" because of the high number of exceptions and requirements to be verified to reach a conclusion, but tax rules are nevertheless quite easy to represent because they do not need any particular work of interpretation by a legal expert.

On the other hand, the PIL domain contains a very high number of concepts which lack a definition set by lawmakers, and which may therefore be considered as "open texture" concepts (as defined in legal philosophy by Hart [3] and discussed in relation to legal information systems by Bench-Capon & Visser [4]). Their meaning should be identified by searching in multiple domain contexts and by taking into account case law and doctrinal interpretations.

Despite the difficulty and complexity of representating PIL, the results present in this report show how potentially any field of law may be represented in the form of rules, using the adopted methodology – with opportune adaptations – and transposed onto a platform such as InterLex.

References

[1] M. U, Brussels I Regulation (European Commentaries on Private International Law), Sellier European law publishers, 2007.

[2] C. G, Rome Regulations: Commentary on the European Rules of the Conflict of Laws, Wolters Kluwer Law & Business, 2015.

[3] H. Hart, The Concept of Law, Oxford: Clarendon Press, 1961.

[4] T. J.M. Bench-Capon and P.R.S. Visser, «Open texture and ontologies in legal information systems,» in *Database and Expert Systems Applications, 1997. Proceedings., Eighth International Conference, DEXA'97. Proceedings. IEEE.*

Appendix

Complete Prolog rule base:

prolog_rules.docx