Lesson 7_Modelling_Norms

Modelling norms

This lesson will deal with the presentation of legal norms in form of rules.

The idea that the law can be **represented** and made **computable** is more than half-century old. In fact, at the beginning of AI there was already this idea that one of the application of symbolic AI would have been the legal domain. This effort has brought to the development of man made models of the law in a way that computer systems can reason about it.

There are two main steps in the development of computable law:

1. The first step concerns on how to **model** and **formalize** the law possibly with a logic formalism. This step takes as input **legal sources**, cases, concepts and all the theories elaborated by the experts that contribute to create taxonomies. Once you have found a way, through the use of **logic programming** and **knowledge representation**, to translate all these things in a computer system, as a result, you will have a **computable model**;
2. Once you have a computable model of the law, it can be used as input in the process of moving from the knowledge encoded in the system to the provision of answers, definition of legal qualifications and activities to support legal decision making. The kind of processes involved in this phase are forward and backward rule chaining, deduction, defeasible reasoning, etc.

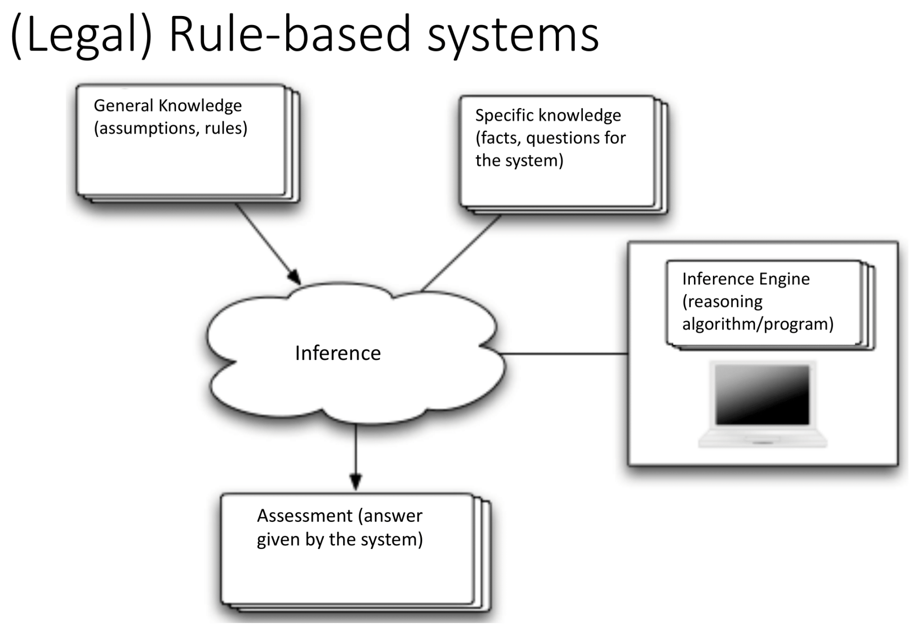
Knowledge Representation

> Knowledge representation is the application of logic and ontology to the task of constructing computable models for some domain.

According to this definition, on one side we have **logic** that provide us the formal structure and rules of inference, then we have **ontologies**, in particular formal ontologies, that defines the kinds of things that exist in the limited domain we want to present, and their relationship; then everything should be made **computable** by finding a way to implement logic and ontology into computer systems.

For doing this, we use, in the symbolic approaches, **declarative programming languages**, like Prolog, in which

- the program consists of logical statements, expressing the knowledge about the domain in terms of known facts and relationships;
- the program executes by searching for proofs of the statements.


(Legal) Rule-based systems
General Knowledge (assumptions, rules)
Specific knowledge (facts, questions for the system)
Inference Engine (reasoning algorithm/program)
Inference
Assessment (answer given by the system)

This type of approach has been used to develop a specific kind of application in symbolic AI which is called the rule-based system and, in particular, in legal domain, the **legal rule-based system**. In this case we need a way to transfer into the computer system a **general knowledge** about the legal domain we want to represent including some assumptions, legal rules. We need also a way to transfer **specific knowledge** of a particular case of a particular legal scenario into the system (facts that correspond to our case, questions that we want the system to provide an answer). Given that in the system we have an **inference engine** (a way to reason about this knowledge), through a process of inference, we get to a result produced by the system.

Legal applications

Looking at the legal domain, beginning from the 80s, a number of researchers had implemented working systems based on manually created logical representations of rules and, in particular, an important contribution to the domain was the work developed by Sergot and Kowalski concerning the British Nationality Act.

Nowadays, rule base systems are used in the legal domain for legal analysis and automated legal assessment. For a number of reasons, not only technological but also to the way the law is produced, this systems have been particularly successful in the anglosaxon context (e.g US, UK and Australia).

Today we have many applications, in particular in public administration (system helping people in assesstments concerning taxes, welfare and system linked with jurical procedures) and in business application.

Why the law has been considered, from the beginning, one of the best testing bed for this kind technology of the symbolic application?

Because we can look at the law as a system where we have a lot of propositions that can be reduced to a set of conditional statements in the form "**if…then**" and therefore we can see much of the legal reasoning as the application of such rules. The idea is that legal rules, seen as conditional statements, are usually shaped in a way to connect abstract provision of facts to a legal effect. For example:

```
if a person Y commits the crime X,
   then Y shall be punished with sanction Z.

If X buys the good Z from Y,
   then X shall pay to Y the price of Z.

If X was born in the territory of State S,
   then X is a citizen of S.
```

There are many ways, then, to move from logical representation to a computable model: prolog, ruleml and various commercial solutions developed by may companies like IBM, Oracle.

The basic scheme of a legal norm represented in the form of conditional statement is made up of:

- **Premises**: where you have
  - the **rule** itself which is a general rule in which you would usually put the universal quantifier

    If ***x*** buys ***z*** from ***y***, then ***x*** shall pay to ***y*** the price of ***z***
  - a **specific fact** that is an instance where you are going to apply the general rule

    Mary buys a ***car*** from ***John***

- **Conclusion**: where you have the legal effect

  Mary shall pay to ***John*** the price of the ***car***

Legal knowledge representation: issues and challenges

More broadly we can say that if we think of issues concerning the representation of legal knowledge, norms and ethical reasonings there are some specific issues concerning the nature of these norms that are very relevant and may create issues and challenges in the representation.

- **Ambiguity**: legal rules may be ambiguous;
- **Vagueness**: also partially related to ambiguity, this has to do with the meaning and connected to particular words or terminology used by the law;
- **Rigidity**: which is the problem that logic representation tends to be rigid by fixing the meaning in predefined symbolic structure that have difficulties in preserving one of the features of the law that is providing a solution, using possibly a set of generic rules, to all the facts happening in the reality. So, it's true that laws are written and fixed in some ways, but they must be used with flexibility by judges in order to provide always a solution in legal cases. This means that there are no cases where there is no solution and in some situation judges have to stretch the interpretation of the legal rule to cover things that possibly were not supposed to be covered by that law;
- **How to represent deontic positions**: that is the idea to encode in some ways the expressivity of things like obbligation, permission or the fact that something is prohibited. That's because the law is not concerned with observing the reality as it is but as it should be;
- **How to enable temporal reasoning**: another big problem that may have a huge impact in the representation of the law is the problem of time. In the case of laws you need to take into account temporal reasoning in rules under different perspectives. Moreover, if such rule is the expression of a legal norm which is part of a legal system, you have to take into account not only the internal times of the rule but also whether or not such rule is a rule that is enforced in the legal system because there may be a time in which such rule has been unacted, entered into force or abrogated.
- **How to deal with conflicting legal rules**: another problem is that most of the time legal rules conflict each other. This phenomena happen many times in the

legal system because there can be laws produced by different authorties like national laws conflicting with Constitution, regional laws conflicting with national ones and so on. So, we need to deal with conflicting legal rules and/or rules that can be be excluded from being applicable by other rules. Lawyers have developed many ways to resolve conflicts, we can have metarules expressing priorities, for example:

  - *lex posterior*: newest rules will prevail the older ones;
  - *lex specialis*: the rule that is meant to solve a specific problem would prevail over a more general rule applicable;
  - *lex superior*: the rule having an higher status in the level of legal sources would prevail over rules at lower levels (a rule contained in the italian Constitution would prevail on every other rule in the italian legal system).

  Then, we also have the problem of the feasibility of the rules: not only we can have rules that conflict with each other but also rules that exclude, from being applicable, other rules;

- **How to manage reification**: the problem of conflicts between rules and the problem of temporal reasoning bring to the issue of how to manage reification, whenever rules representing legal norms need to be treated as object with properties by other rules (e.g metarules meant to solve conflicts between rules);
- **How to maintain isomorphism between source text and representation**: this is a pratical problem on one side but also a legal requirement in the sense that in the law the official text pubblished by the legislator is the text which is legally binding. So, the lawyers attach a kind of legal property to the quality of such text and the exact sequence of words, despite them of being vague and ambigous or erroneous, is still what the law exactly says.

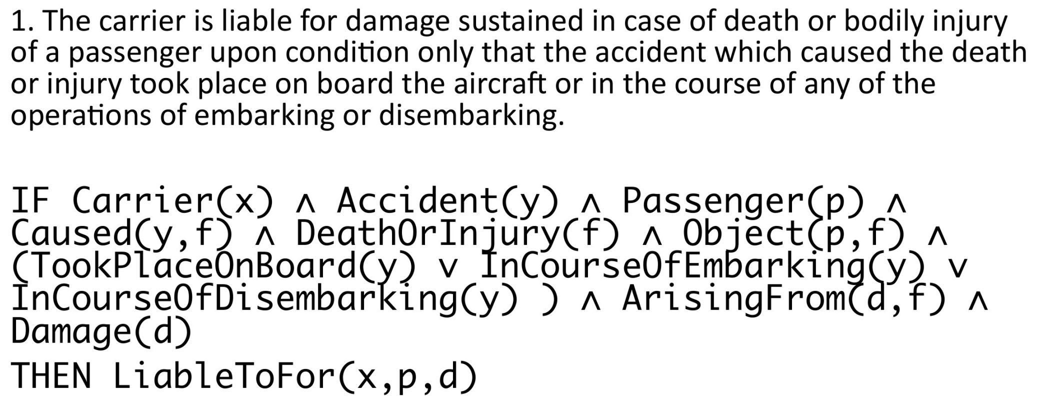
  For the sake of efficiency, most of the time lawyers are forced to deconstruct and rebuild them in a more efficient logical form. This is a big problem because whenever you do this you are introducing the risk that the logical representation will not be faithful to the original text and moreover it will be much more complicated to **maintain the logical structure** and **introduce modifications** (in some legal domain the laws are updated at least once a year) once it is separated from the original text.

  Another problem, linked to this, is the **explaination**. In fact, the symbolic representation of the law is transparent and can provide a good explaination of their decisions. However, in a legal context, the kind of explaination that it is expected from a lawyer is an explaination grounded to the source text of the law. So, by having an isomorphic representation, the correspondance between the logical representation and the source binding legal text can be easily traced;

- **Other more practical issues**: the reason why many of this system never moved from academic prototype to real world application is the fact that the effort, required to **elicitate** (extract and distill) the knowledge from the books or human expert, to **represent** and **update** it, is so much that is not worth when looking at the benefits of such systems. This is also the reason why such systems have been successful only in specific domains, in paticular those domains in which you have a limited number of previously seen issues. For example, tax law is a very good domain for representation because the complexity is given by the incredibly high number of chainings between rules that taken in isolation are very easy to be represented. So, there is a complexity but it is given by the way rules are chained together and not by problems of interpretation or linked to the vagueness of the text.

Example of logical representation of a norm

To give an idea of this kind of representation, this is an example of a logic representation dealing with the liability of air carriers in case of accident to passengers.



1. The carrier is liable for damage sustained in case of death or bodily injury of a passenger upon condition only that the accident which caused the death or injury took place on board the aircraft or in the course of any of the operations of embarking or disembarking.

```
IF Carrier(x) ∧ Accident(y) ∧ Passenger(p) ∧
Caused(y,f) ∧ DeathOrInjury(f) ∧ Object(p,f) ∧
(TookPlaceOnBoard(y) ∨ InCourseOfEmbarking(y) ∨
InCourseOfDisembarking(y) ) ∧ ArisingFrom(d,f) ∧
Damage(d)
THEN LiableToFor(x,p,d)
```

Example of ambiguity

Then that's an example of ambiguity in the legal domain. The article presented is the legal provision sanctioning hackers.

- **Art. 615/ter of Italian criminal code, (unauthorised access to a computer system):**

"Whoever enters a computer or telecommunication system which is protected by security measures or remains in such system against the will of the person who is entitled to exclude him, shall be punished with detention up to three years"

If [a: the individual enters the computer or telecommunication system]
and [b: the computer or telecommunication system is protected by security means]
or [c: the individual remains in the computer or telecommunication system]
and [d: there is the contrary will of the person who is entitled to exclude the individual]
then [e: the individual shall be punished with detention up to three years]

The problem, here, is that despite the fact we understand exactly what are the pieces of the puzzles, we have difficulties in putting them in the right order and with the right connectives to build the conditional statement. This has to do with the fact that some terms in the natural language or the way the statement is expressed are ambigous in themselves.

Here, in particular, the ambiguity is caused in the use of "**such a system**" because when used in the text it is not clear exactly what system it is refferred to (does it mean every computer or telecommunication system or those computer and telecommunication systems which are protected by security measures?).

Another point is in the use of "**against the will of the person**" beacuse the fact that there must be the contrary will of the person to be considered a crime is connected only to the act of remaining or is it also connected to the act of entering in the system?

Depending on how you deal with these ambiguities, three different logical representations can be developed:

- IF {(a AND b) OR (c AND d)} THEN e
- IF {(a OR c) AND (b AND d)} THEN e
- IF {(a OR c) AND (b OR d)} THEN e

So, it is clear that, depending on the reconstruction the judge produce, during a criminal procedure with the same facts, they can choose to conclude that a person is or not a criminal.

In the history of the application of this law, it has been seen a trend of judges converging in the first of the three interpretations but at least one judge has chosen one of the other two.

Example of vagueness

This is a fundamental problem not only in legal reasoning, but also in moral reasoning. It is a problem whenever we have situations in which there are terms that are difficult to enclose with a clear border. The definition of vagueness given by **Hart**, a famous philosopher, is:

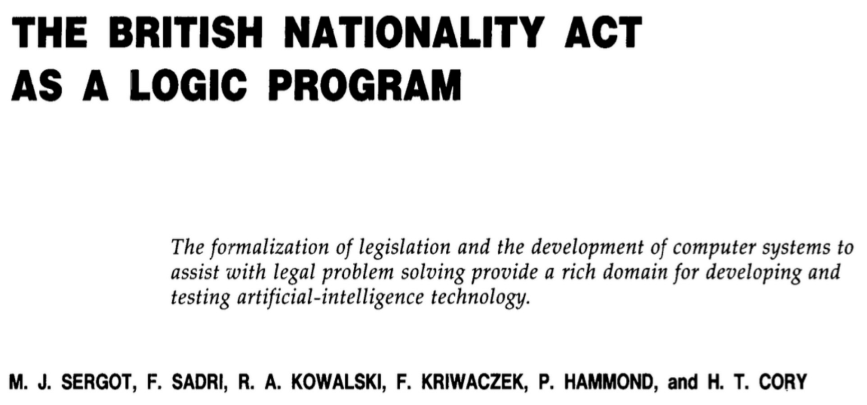
> …All rules involve recognizing or classifying particular cases as instances of general terms, and in the case of everything which we are prepared to call a rule it is possible to distinguish clear central cases, where it certainly applies and others where there are reasons for both asserting and denying that it applies. Nothing can eliminate this duality of **a core of certainty and a penumbra of doubt** when we are engaged in bringing particular situations under general rules. This imparts to all rules a fringe of vagueness or '**open texture**'…

According to Hart, whenever we engage a rule we have this problem of vagueness and we have to cope with it. But also very simple words, that theoretically seems to not create debate, may open incredible problems of vagueness. That's the case of a famous example from Hart in which at the entrance of a park there is sign stating:

> No vehicles allowed in the park

Apart from the trivial vehicles like cars, motorbikes, coaches, etc. there is a penumbra of cases (e.g bikes, skateboards, horses, trolleys, ambulances, police cars, monument representing vehicles) in which, depending on the interpretation of the law, we can choose to expand or reduce the kind of things that are covered by it.

Paper: The British Nationality Act as a Logic Program


THE BRITISH NATIONALITY ACT
AS A LOGIC PROGRAM

The formalization of legislation and the development of computer systems to assist with legal problem solving provide a rich domain for developing and testing artificial-intelligence technology.

M. J. SERGOT, F. SADRI, R. A. KOWALSKI, F. KRIWACZEK, P. HAMMOND, and H. T. CORY

This is a very important paper written in 1986 and it is a report on a research for representing the British Nationality Act of 1981 using Prolog.

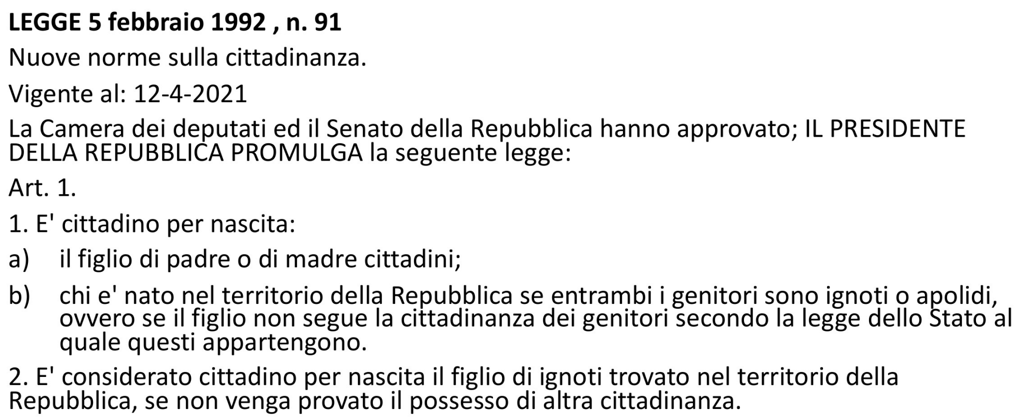
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5f8a34c2-57d4-4c0e-b603-cf004b33c225/paper_Sergot_Kowalski.pdf

❗ He basically reads the highlighted parts of the paper

Modelling the Italian Nationality Act

Now we are going to make the same exercise that Sergot and Kowalski made in the paper but with the Italian Nationality Act.

Original text:


LEGGE 5 febbraio 1992 , n. 91

Nuove norme sulla cittadinanza.

Vigente al: 12-4-2021

La Camera dei deputati ed il Senato della Repubblica hanno approvato; IL PRESIDENTE DELLA REPUBBLICA PROMULGA la seguente legge:

Art. 1.

1. E' cittadino per nascita:

a) il figlio di padre o di madre cittadini;

b) chi e' nato nel territorio della Repubblica se entrambi i genitori sono ignoti o apolidi, ovvero se il figlio non segue la cittadinanza dei genitori secondo la legge dello Stato al quale questi appartengono.

2. E' considerato cittadino per nascita il figlio di ignoti trovato nel territorio della Repubblica, se non venga provato il possesso di altra cittadinanza.

English version:

> Article 1
>
> 1. The following shall be citizens by birth:
>   a) any person whose father or mother are citizens;
>   b) any person who was born in the territory of the Republic, either where both parents are unknown or stateless, or where he or she does not acquire his or her parents' citizenship according to the law of the State to which the latter belong;
>
> 2. Any person who is found in the territory of the Republic, whose parents are unknown, shall be deemed a citizen by birth, where their possession of any other citizenship cannot be proven.

Software demonstration with Prolog

```
citizen(A):- (father(B, A); mother(B,A)), citizen(B).
```

```
citizen(A):-
      born_in_republic(A),
      father_ineligible(A),
      mother_ineligible(A).

father_ineligible(A):-
      father(unknown, A);
      father(stateless, A).

mother_ineligible(A):-
      mother(unknown, A);
      mother(stateless, A).
```

Oracle Policy Automation

It is a suite of tools that help to write rules quickly with Microsoft Word and is provided with a linguistic parser able to analize the syntactic structures of propositions in order to identify logical components. The rules are, then, translated into an XML format and the inference engine is able to reason on these ones. The system provides also a very good out of the box web user interface that makes very easy the creation of interfaces for querying and providing justification of results.

Software demonstration

Interlex

InterLex is an EU funded project that aims at developing an online platform to provide information, decision support and training on private international law. It addresses the identification of the legal system having jurisdiction and of the national law to be applied to a specific case as well as the retrieval of relevant legal materials. For this project an implementation of Prolog has been used, which gave the possibility to expand the basic implementation with external modules like CLP or meta-interpreter.

Example: Article 4.1

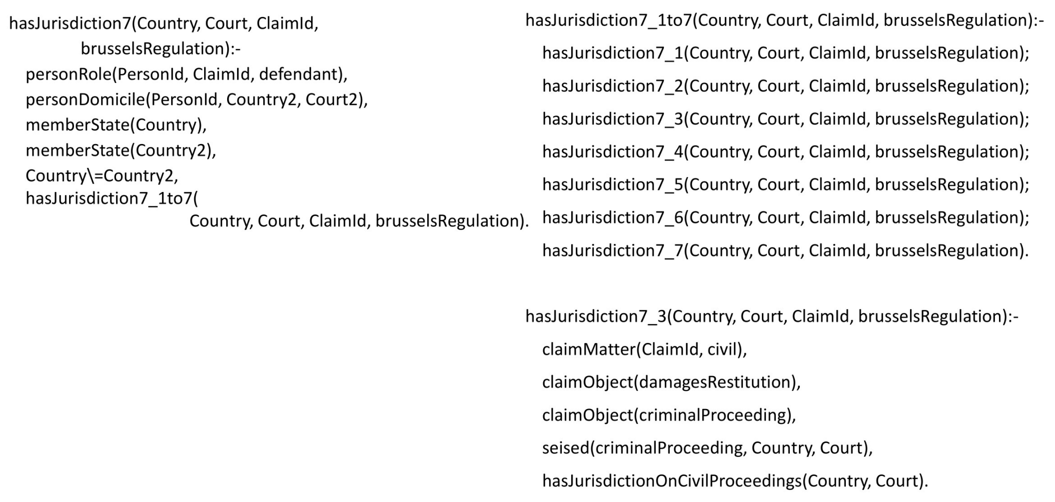
Subject to this Regulation, persons domiciled in a Member State shall, whatever their nationality, be sued in the courts of that Member State.

```
hasGeneralJurisdiction(Country, Court, ClaimId,
                       brusselsRegulation):-
    personRole(PersonId, ClaimId, defendant),
    personDomicile(PersonId, Country, Court),
    memberState(Country).
```

Example: Article 7.3

Articles may be also more complex and in this case are divided in multiple different parts.

A person domiciled in a Member State may be sued in another Member State: as regards a civil claim for damages or restitution which is based on an act giving rise to criminal proceedings, in the court seised of those proceedings, to the extent that that court has jurisdiction under its own law to entertain civil proceedings;

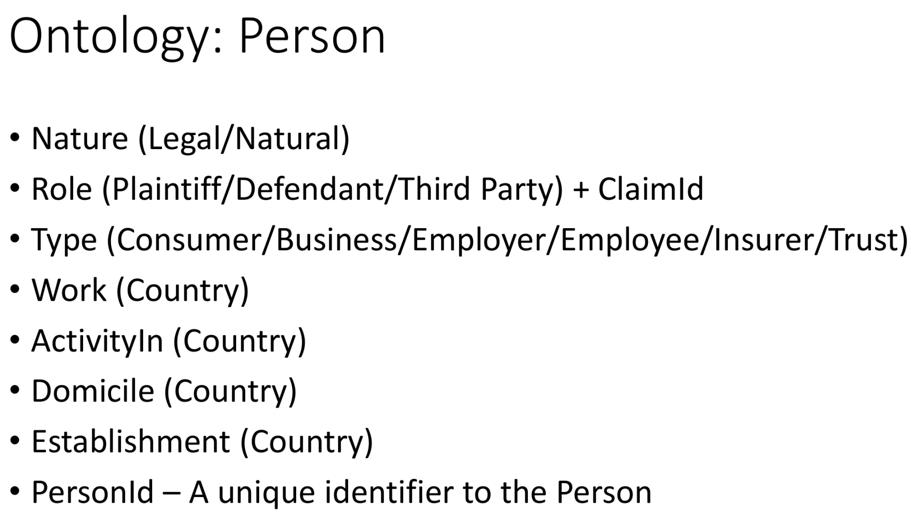


```
hasJurisdiction7(Country, Court, ClaimId,
            brusselsRegulation):-
   personRole(PersonId, ClaimId, defendant),
   personDomicile(PersonId, Country2, Court2),
   memberState(Country),
   memberState(Country2),
   Country\=Country2,
   hasJurisdiction7_1to7(
                         Country, Court, ClaimId, brusselsRegulation).
```

```
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_3(Country, Court, ClaimId, brusselsRegulation):-
   claimMatter(ClaimId, civil),
   claimObject(damagesRestitution),
   claimObject(criminalProceeding),
   seised(criminalProceeding, Country, Court),
   hasJurisdictionOnCivilProceedings(Country, Court).
```

In order to model complex rules we firstly have to define ontologies to design what are the main actors of our system and what are their properties.



Ontology: Person

- Nature (Legal/Natural)
- Role (Plaintiff/Defendant/Third Party) + ClaimId
- Type (Consumer/Business/Employer/Employee/Insurer/Trust)
- Work (Country)
- ActivityIn (Country)
- Domicile (Country)
- Establishment (Country)
- PersonId – A unique identifier to the Person

The advantages of using Prolog are that:

- rules are compact and readable;
- the logical structure of rules matches the legal text;
- exceptions can be easily introduced;
- the closed-word assumption is implemented in the system (what does not hold is assumed to be false);
- the system is capable of explaining how it arrived to a certain solution.

Software demonstration