# A methodology to create legal ontologies in a logic programming information retrieval system

José Saias and Paulo Quaresma

Departamento de Informática,
Universidade de Évora,
7000 Évora, Portugal
jsaias|pq@di.uevora.pt

**Abstract.** Legal web information retrieval systems need the capability to reason with the knowledge modeled by legal ontologies. Using this knowledge it is possible to represent and to make inferences about the semantic content of legal documents.
In this paper a methodology for applying NLP techniques to automatically create a legal ontology is proposed. The ontology is defined in the OWL semantic web language and it is used in a logic programming framework, EVOLP+ISCO, to allow users to query the semantic content of the documents. ISCO allows an easy and efficient integration of declarative, object-oriented and constraint-based programming techniques with the capability to create connections with external databases. EVOLP is a dynamic logic programming framework allowing the definition of rules for actions and events.
An application of the proposed methodology to the legal information retrieval system of the Portuguese Attorney General's Office is described.

## 1 Introduction

Modern legal information retrieval systems need the capability to represent and to reason with the knowledge modeled by legal ontologies. In fact, the creation of ontologies allow the definition of class hierarchies, object properties, and relation rules, such as, transitivity or functionality. Using this knowledge it is possible to represent semantic objects, to associate them with legal documents, and to make inferences about them.

OWL (Ontology Web Language) is a language proposed by the W3C consortium (http://www.w3.org) to be used in the "semantic-web" environment for the representation of ontologies. This language is based on the previous DAML+OIL (Darpa Agent Markup Language - [1]) language and it is defined using RDF (Resource Description Framework - [2]).

In this paper a methodology to automatically create an OWL ontology from a set of legal documents is proposed. The methodology is based on the following steps:

- Definition of an initial top-level ontology;

– Identification of concepts referred in the legal documents and extraction of its properties;
– Identification of relations between the identified concepts;
– Creation of an ontology using the identified concepts and relations;
– Merge of the created ontology with the initial ontology;

In the first step, an already existent top-level legal ontology was chosen. At present we are using the legal ontology from the Portuguese Attorney General's Office, consisting of around 6,000 classes and having around 10,000 relations [3]. However, other top-level ontologies could be used, such as, the DOLCE proposal [4] or the FOLaw and LRI-core proposal [5] used in the context of the IST programs E-POWER and E-COURT [6]. In the future, we expect to use the results of the e-Content project LOIS – Lexical Ontologies for legal Information Sharing, which aims to create an european-wide top-level legal ontology.

In the second step, identification of concepts and its properties, several natural language processing techniques are used, namely, a syntactical parser, and a semantic analyzer able to obtain a partial interpretation of the documents. As it will be described in detail, the semantic representation allows the identification of the set of concepts that are referred in the documents and the extraction of some of their properties.

In the third step, identification of relations between concepts, an unsupervised method for acquiring word classes and relations is used [7, 8]. This method, which has some similarities with the work of [9], allows the identification of related and more specific concepts (subclasses). Starting from parsed documents, a subcategorisation analysis is performed and, for each word, subcategorisation patterns are extracted. Finally, a statistical analysis is performed identifying clusters of words with similar subcategorisation patterns.

In the fourth step, the results of the previous two steps are integrated in an ontology: concepts with their properties are new classes; class hierarchies and relations are created accordingly with the statistical analysis of subcategorization patterns (several examples will be shown in the following sections).

Finally, in the fifth step, the initial top-level ontology is merged with the new one. The proposed strategy is to search for common concepts in the two ontologies and to merge the ontologies via these concepts. New classes are inserted into the top-level ontology using the information from the semantic analyser (animal, human, action, . . . ).

At this stage it is important to point out that the proposed methodology is based on a bottom-up approach for the definition of the legal low-level concepts: it allows the identification of the concepts and some of the relations, but additional work will be necessary to fully integrate these concepts with the upper legal ontology. We do not intend to propose any kind of standard for legal ontologies; our aim is to define a methodology to automatically create a base ontology from a specific set of legal documents.

As referred, this work has some relations with the proposal of Lame aiming to identify components of legal ontologies from the analysis of legal texts [9]. However, we believe our proposal has a more ambitious goal: the identified components are used to create an ontology and the initial documents are enriched with instances of this ontology. This process allows the definition of semantic web agents able to query the semantic content of these documents.

As stated before, after the creation of legal ontologies expressed in OWL, documents are enriched with instances of legal classes.

Then, a logic programming based framework is used to support inferences over the ontology. The logic programming framework is based on ISCO [10] and EVOLP [11].ISCO is a new declarative language implemented over GNU Prolog with object-oriented predicates, constraints and allowing simple connections with external databases. EVOLP is a dynamic logic programming language that is able to describe actions and events, allowing the system to make inferences about events, user intentions and beliefs and to be able to have cooperative interactions.

This logic programming framework seems to be quite adequated to represent and to make inferences over OWL ontologies. In fact, recent advances in the semantic web technology support this claim: some partners in the RuleML workgroup (http://www.ruleml.org) are adopting logic programming as its inference engine and there already exists a translator from RDFS to Prolog [12]. Moreover, in the scope of this work, a translator of a subset of OWL to Prolog was also built (correct accordingly with the OWL formal semantic description).

However, other inference engines could be chosen and used to answer queries about the legal knowledge conveyed by the documents. One possible option might be to use the results of the Mandarax project (http://mandarax.sourceforge.net), which already supports RuleML.

Section 2 describes the proposed architecture. Section 3 describes the methodology for the creation of the ontology, namely, the natural language processing techniques used to process the documents. Section 4 describes the NLP techniques used to create the OWL instances associated with each document. Section 5 describes ISCO, the basic logic programming framework. Section 6 describes EVOLP, the dynamic logic programming framework defined over ISCO and Prolog. Section 7 describes the interaction manager and section 8 provides a simple example. Finally, in section 9 some conclusions and future work are pointed out.

## 2 Architecture

The system's architecture is based on several independent and modular processes. Figure 1 shows graphically these processes and their relations.

The architecture may be divided in three major modules:

- Inference of an adequate OWL ontology of classes;
- Inference of OWL instances and document enrichment;
- Inference engine.

The first module, inference of an adequate OWL ontology of classes, receives as input a top-level ontology and a set of legal documents. After a syntactical and semantical analysis, it obtains a partial semantic representation (a DRS – Discourse Representation structure [13]) for each sentence. From the DRS of

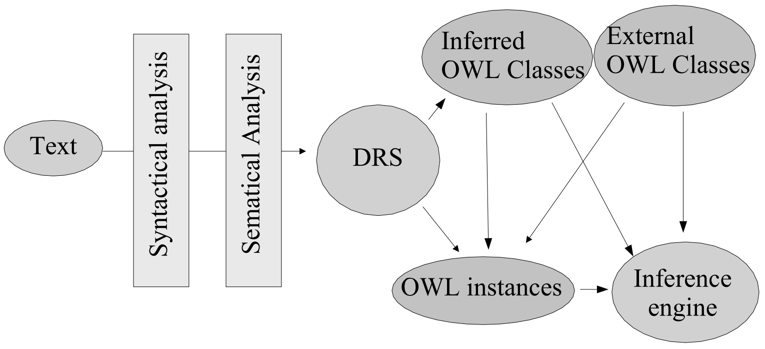

**Fig. 1.** System's architecture

each sentence, noun expressions and verbs are extracted, and they are used to define legal classes. These classes will be clustered, classified, and merged with the initial top-level ontology (section 3 describes this step in detail).

The second module, inference of OWL instances, receives as input the DRS of each sentence and the inferred OWL ontology from the first module. With this information, and using an abductive inference mechanism, OWL instances are inferred. This step is usually called pragmatic interpretation of natural language sentences (section 4 describes these processes in more detail).

Finally, OWL classes and OWL instances are used by an inference engine, based in a logic programming framework, in order to answer queries about the semantic content of documents (sections 5, 6 and 7 describe the logic programming framework).

## 3 OWL ontology creation

In order to be able to deal with documents from different domains, a methodology to automatically create basic ontologies of concepts is proposed. This methodology allows the definition of a base ontology with the relevant concepts with some inferred relations. After having defined this ontology, it may be necessary to develop manual work by human experts in order to fully organize the set of extracted concepts.

The methodology is based on the following steps:

– Definition of an initial top-level ontology;
– Identification of concepts referred in the legal documents and extraction of its properties;
– Identification of relations between the identified concepts;
– Creation of an ontology using the identified concepts and relations;
– Merge of the created ontology with the initial ontology;

## 3.1 Top-level ontology

As referred in section 1, an already existent top-level legal ontology was chosen: the legal ontology from the Portuguese Attorney General's Office, consisting of around 6,000 classes and having around 10,000 relations [3]. As and example of some concepts in this ontology we have:

– Tribunal *Court*; properties: name, address, ...
– Tribunal Militar *Military Court* – subclass of Tribunal
– Supremo Tribunal *Supreme Court* – subclass of Tribunal

This legal ontology was merged with a general top-level ontology of concepts defined by Eckhard Bick in the VISL project [1] [14], which has around 150 top concepts: animal, human, place, vehicle, concrete object, abstract object, food, ....

## 3.2 Identification of concepts and properties

The methodology to automatically identify the concepts and the properties referred in the documents is based on the output of natural language processing tools:

– Text syntactical parsing. The documents are analyzed by the syntactical parser PALAVRAS developed by E. Bick. This parser is available for 21 different languages,including Portuguese.
– Partial semantic analysis.
– Entities extraction. From the semantic analysis output, entities and properties are extracted and represented by ontology classes.

**Syntactical analysis** The parser developed by E. Bick is based on the Constraint Grammars [15] formalism and covers a major portion of the Portuguese sentences. However, its output is in a non-standard format and it was necessary to transform it into a structured form, like XML and Prolog terms. A translation tool from the VISL output into XML and Prolog terms was developed and it is available to the VISL users (a detailed description of this tool was presented in [16]).

As an example, suppose the following sentence:

O bombeiro Manuel salvou a criança. *The fireman Manuel saved the child.*

This sentence has the VISL output:

```
STA:fcl
SUBJ:np
=>N:art('o' M S)        O
=H:n('bombeiro' M S)    bombeiro
==N<:prop('Manuel' M S) Manuel
```

[1] http://visl.hum.sdu.dk/visl

```
P:v-fin('salvar' PS 3S IND)     salvou
ACC:np
=>N:art('a' F S)        a
=H:n('criança' F S)     criança
```

As it can be seen, the subject, predicate and direct object were correctly parsed.

**Semantic analysis** Each syntactical structure is translated into a First-Order Logic expression. The technique used for this analysis is based on DRS's (Discourse Representation Structures [13]). The partial semantic representation of a sentence is a DRS built with two lists, one with the rewritten sentence and the other with the sentence discourse referents.

At present, we are only dealing with a restricted semantic analysis and we are not able to handle every aspect of the semantics: our focus is on the representation on concepts (nouns and verbs) and the correct extraction of its properties (modifiers, agents, objects).

From the XML structure, using XSL transformations, it is possible to obtain the semantic representation of each sentence.

The semantic representation of the example presented in the previous subsection is:

sentence(doc1, [fireman(A), name(A,'Manuel'), child(B), save(A,B)], [ref(A), ref(B)]).

This structure represents an instance of a fireman $A$, named 'Manuel', and an instance of a child $B$ which are related by the action *to save*.

A general tool able to obtain similar semantic partial representations for every sentence was developed and it was applied to the full set of legal documents of the Portuguese Attorney General's Office (7000 documents).

**Entities extraction** From the sentence semantic representation, entities are extracted and they are the basis for the creation of an ontology of concepts. In fact, for each new concept, a new class, subclass of a top-class 'Entity', is created.

On the other hand, from the output of the semantic analyser it is possible to identify some potential class properties:

– Modifiers, such as adjectives, are candidates to be properties of nouns;
– Direct objects of transitive verbs are candidates to be properties of the associated verbs;

For instance, for the expression *the black cat* it is possible to identify *color* as a property of *cat*, because it is known that *black* is an instance of a color (from the correspondent semantic tag in the dictionary).

In the referred example it would be possible to extract the following entities:

– bombeiro *fireman*, with a property: 'name'

– salvar *to save*
– criança *child*

## 3.3 Identification of relations between classes

As it was shown in the previous sub-section, the identification of concepts does not allow the creation of relations, hierarchical or others, between them.

Our approach is to use an an unsupervised method for acquiring word classes and relations [7, 8]. The goal is to learn, for each word, what kind of modifiers and what kind of heads it subcategorises. For instance, the word *republic* may appear as an head of a noun phrase, such as *republic of Ireland*, *republic of Portugal*, or as a modifier, like *president of the republic*, *government of the republic*. The obtained subcategorisation patterns are clustered into classes and relations are extracted.

Using this approach it is possible to identify hierarchical relations, such as the existent between *republic* and *republic of Portugal* and also to identify other semantic relations, such as the ones between *lei – law* and *norma – norm*. The strategy is to use statistical analysis to identify clusters of words with similar subcategorisation patterns (words which have similar modifiers and heads).

As methodology, we start from the parsed documents and, for each word, subcategorisation patterns are extracted and clusters and relations are identified. A detailed description of the methodology is described in [8].

Note that this approach has some limitations and it is not able to identify correctly what kind of relations exist between two concepts. For instance, two related concepts may be synonyms or the opposite. A more deep knowledge-aware approach is needed to handle these kind of problems.

The inferred relations are used to create an hierarchy of classes in the ontology and to link them via a *related* relation.

## 3.4 Creation of the ontology

In the fourth step of the methodology, the results of the previous two sub-sections are integrated into a new ontology:

– Concepts with their properties are new classes;
– Class hierarchies and relations are created accordingly with the results of the previous sub-section;

For instance, using this approach to the previous example, *republic of Portugal* will be a sub-class of *republic*.

## 3.5 Merge of the ontologies

Finally, in the fifth step, the initial top-level ontology is merged with the new one.

In this process new classes are inserted into the top-level ontology using their names and information from the semantic analyser:

– If a class exists with an equal name in the top-level ontology, then the two classes are merged;
– Otherwise, a search is made in the top-level ontology for a class with semantically compatible information and the new class is created as a sub-class of the existent one.

For instance, if a new class named *fireman* is classified to be a *human* concept by the NLP analysers, then the new class will be a sub-class of the *human* top-level concept.

The overall strategy is to search for common concepts in the two ontologies and to merge the ontologies via these concepts.

## 4 OWL instances creation

After having defined an ontology of classes, it is necessary to extract and to represent instances of those classes and to associate them with documents.

The proposed methodology tries to infer instances of those ontologies using the following three steps:

– Translation of the OWL ontologies into a logic programming form;
– Definition of logic programming rules allowing the inference of instances;
– Generation of OWL instances.

**OWL translation** The first step, translation of OWL ontologies into Prolog, was implemented in Java and it creates a Prolog term for each OWL class, subclass, or property. The translation of this subset on OWL is correct accordingly with the OWL formal semantic description [17].

For instance, suppose there exists a definition in OWL for a class *citizen* and for a sub-class *military.* After the translation, we'll have:

```
class(citizen, 'external.owl#citizen').
class(military, 'external.owl#military').

subclass('external.owl#military', 'external.owl#citizen').
```

Moreover, suppose class *military* has a property of having a *rank*, which can have one of several possible values: general, colonel, . . .

In this situation, we'll have the following Prolog terms:

```
property(rank, 'external.owl#rank',
                'external.owl#military').

hasPossibleValue('external.owl#rank', general).
hasPossibleValue('external.owl#rank', colonel).
```

**Prolog rules** In the second step of this methodology, logic programming rules are defined allowing the inference of instances from the DRS representation of each sentence and the Prolog representation of the ontology.

One of these rules allows the inference of class and properties from values:

```
infer(Value, Class, Property) :-
                hasPossibleValue(PropertyURI, Value),
                property(Property, PropertyURI, ClassURI),
                class(Class, ClassURI).
```

In this LP rule, *Value* is the name of an entity (input) and *Class* and *Property* are identifiers of classes and properties that may have this value (output).

For instance, the sentence

```
The colonel saved the child.
```

has the following DRS form:

```
sentence(d1, [colonel(X), child(Y), save(X,Y)],
             [ref(X),ref(Y)]).
```

From this DRS form and, using the Prolog rules, it is possible to infer the following new form (because *colonel* is a possible value for the *rank* property of the *military* class):

```
sentence(d1, [military(X), rank(X,colonel), child(Y),
              save(X,Y)], [ref(X),ref(Y)]).
```

This process is usually called, in the natural language processing field, pragmatic interpretation of sentences and it can be seen as an abductive process where properties (antecedents) are inferred from values (consequents) [18].

Similar approaches can be applied to capture different natural language sentences characteristics.

**OWL generation** In the third step, the results of the pragmatic interpretation of each sentence are transformed into correspondent OWL instances. For instance, for the last example of the previous sub-section, the following OWL instances would be created:

```
<pgr:Military rdf:ID="m11">
   <pgr:rank rdf:resource="external.owl#Colonel"/>
   <pgr:belong rdf:resource="external.owl#Army"/>
</pgr:Military>

<pgr:Child rdf:ID="c2">
</pgr:Child>

<pgr:ToSave rdf:ID="s5">
   <pgr:subject rdf:resource="#m11"/>
   <pgr:object rdf:resource="#c2"/>
</pgr:ToSave>
```

These instances define and relate a military (colonel and from the army), a child, through the instance of the action *to save*.

As a final result of this step, every document is enriched with the OWL instances obtained from the pragmatic interpretation of its sentences.

## 5 ISCO

In this section, the logic programming framework that is going to be used as the inference engine for answering queries about the semantic content of documents (OWL instances) is briefly described.

ISCO [10] is a logic based development language implemented in GNU Prolog that gives the developer several distinct possibilities:

– It supports Object-Oriented features: classes, hierarchies, inheritance.
– It supports Constraint Logic Programming. Specifically, it supports finite domain constraints in ISCO queries.
– it gives a simple access to external relational databases through ODBC. It has a back-end for PostgreSQL and Oracle.
– It allows the access to external relational databases as a part of a declarative/deductive object-oriented (with inheritance) database. Among other things, the system maps relational tables to classes – which may be used as Prolog predicates.
– It gives a simple database structure description language that can help in database schema analysis. Tools are available to create an ISCO database description from an existing relational database schema and also the opposite action.

Taking these ISCO features into account, a translator from OWL into ISCO class definitions was developed. This translator was applied to every OWL class described in the previous section and, as a consequence, correspondent SQL tables and ISCO classes definitions were obtained. Moreover, each OWL class instance was transformed into an SQL table row and an ISCO logic programming fact. As an example, the *toSave* presented previously is translated into the following fact:

```
toSave(ID=s5, subject='#m11', object='#c2').
```

For each defined class a set of Prolog predicates implementing the four basic operations are created: query, insert, update and delete.

Variables occurring in queries are mapped to SQL and may carry CLP(FD) constraints, which will be expressed in SQL, whenever possible. For example, suppose variable `X` is an FD variable whose domain is `(1..1000)`, the query

$$\texttt{document(number = X, title = Y)} \tag{1}$$

will return all pairs `(X, Y)` where `X` is a document number and `Y` is the document's title. `X` is subject to the constraints that were valid upon execution of the query, ie. in the range 1 to 1000.

ISCO class declarations feature inheritance, simple domain integrity constraints and global integrity constraints.

## 6 EVOLP

As it was described in the previous section, ISCO allows a declarative representation of ontologies and object instances. However, there is also a need to represent actions and to model the evolution of the knowledge.

In [19] it was introduced a declarative, high-level language for knowledge updates called *LUPS* (Language of UPdateS) that describes transitions between consecutive knowledge states. Recently, a new language, EVOLP [11], was proposed having a simpler and more general formulation of logic program updates. In this section a brief description of the EVOLP language will be given. A detailed description of the language and of its formalization is presented at the cited article.

EVOLP allows the specification of a program's evolution, through the existence of rules which indicate assertions to the program. EVOLP programs are sets of generalized logic program rules defined over an extended propositional language $L_{assert}$, defined over any propositional language $L$ in the following way [11]:

- All propositional atoms in $L$ are propositional atoms in $L_{assert}$
- If each of $L_0, \ldots, L_n$ is a literal in $L_{assert}$, then $L_0 \leftarrow L_1, \ldots, L_n$ is a generalized logic program rule over $L_{assert}$.
- If $R$ is a rule over $L_{assert}$ then $assert(R)$ is a propositional atom of $L_{assert}$.
- Nothing else is a propositional atom in $L_{assert}$.

The formal definition of the semantics of EVOLP is presented at the referred article, but the general idea is the following: whenever the atom $assert(R)$ belongs to an interpretation, i.e. belongs to a model according to the stable model semantics of the current program, then $R$ must belong to the program in the next state. For instance, the following rule form:

$$assert(b \leftarrow a) \leftarrow c \tag{2}$$

means that if $c$ is true in a state, then the next state must have rule $b \leftarrow a$.

EVOLP has also the notion of external events, i.e. assertions that do not persist by inertia. This notion is fundamental to model interaction between agents and to represent actions. For instance, it is important to be able to represent actions and its effects and pre-conditions:

$$assert(Effect) \leftarrow Action, PreConditions \tag{3}$$

If, in a specific state, there is the event *Action* and if *PreConditions* hold, then the next state will have *Effect*.

## 7 Interaction Management

The interaction manager is built on the ISCO+EVOLP logic programming framework.

As final goal, we aim to handle the following kind of questions:

- Situations where action A is performed
- Situations where action A is performed having subject S
- Situations where S is the subject of an action

Note that the inference engine needs to be able to deal with the ontology relations. For instance, the question "situations where action A is performed having subject S" means "situations where action A (or any of its sub-classes) is performed having subject S (or any of its sub-classes)".

The interaction manager is composed by the following main tasks:

- Query management
- Interaction management

### 7.1 Query management

The analysis of a natural language query is split in three subprocesses: Syntax, Semantics, and Pragmatics.

**Syntax** As syntactic analyser we are using the analyzer developed by E. Bick and referred previously [14]. The VISL output is translated into Prolog facts by the same translator referred in section 3. This translation can be handled by the same translator because there is a direct relation between the XML structure and the Prolog term structure.

As an example, the following query:

```
Quem salvou crianças?
‘‘Who saved children?’’
```

Has the following syntactical structure:

```
sentence(syn(que(fcl,
     subj(pron_indp('quem','M/F','S','<interr>'),'Quem'),
     p(v_fin('salvar','PS','3S','IND'),'salvou'),
     acc(n('criança','F','P','<H>'),'crianças', '?')))).
```

**Semantics** As referred in section 3, each syntactical structure is translated into a First-Order Logic expression (DRS). The semantic representation of a sentence is a DRS built with two lists, one with the rewritten sentence and the other with the sentence discourse referents. For instance, the semantic representation of the sentence above is the following expression:

```
child(B), toSave(A,B),
```

and the following discourse referents list:

```
A : [ref(A),ref(B)]
```

These structures represent instances of children $B$ related with instances of the *toSave* action.

Note that, at present, we are not able to deal with general unrestricted queries and to translate them from a syntactical into a semantic structure. In fact this a quite complex NLP problem and we have decided to deal only with specific subsets of the Portuguese language, namely, with interrogatives about specific domains.

**Pragmatic Interpretation** The pragmatic module receives the semantic query representation and tries to interpret it in the context of the database information, which was constructed from the translation of the OWL instances into ISCO facts (as described previously in section 5).

In order to achieve this behavior the system tries to find the best explanations for the sentence logic form to be true in the knowledge base. As already referred, this strategy for interpretation is known as "interpretation as abduction" [18].

From the description of the OWL (and ISCO) classes it is possible to obtain the correspondent ISCO query:

```
child(id=B),
toSave(id=C, subject=A, object=B),
```

The interpretation of the ISCO predicates is done by accessing the knowledge base in order to collect (and constraint) all entities identifiers:

```
- $A=_\#(104..109:156..157)$ -- A constrained to all
entities with the desired properties
```

The above expression contains the possible interpretations of the query in the context of the knowledge base.

## 7.2 Interaction Management

The interaction manager has to represent the actions associated with the queries (*informs* or *request*), and to model the user attitudes (intentions and beliefs).

This task is also achieved through the use of the EVOLP language (see [20] for a more detailed description of these rules). For instance, the rules which describe the effect of an inform, and a request speech act are:

$$assert(bel(A, bel(B, P))) \leftarrow inform(B, A, P). \tag{4}$$

$$assert(bel(A, int(B, Action))) \leftarrow request(B, A, Action). \tag{5}$$

These rules mean that if an agent $A$ is informed of a property $P$, then it will start to believe that the other agent believes in $P$; additionally, if $B$ requests $A$ to perform an action *Action*, then $A$ starts to believe that $B$ intends *Action* to be performed.

In order to represent collaborative behavior it is also necessary to model the transference of information between the agents:

$$assert(bel(A,P)) \leftarrow bel(A, bel(B,P)). \tag{6}$$
$$assert(int(A, Action)) \leftarrow bel(A, int(B, Action)). \tag{7}$$

These two rules means that if an agent $A$ believes another agent believes in $P$, then it will start to believe in $P$ (it is a cooperative, credulous agent); moreover, it will also adopt the intentions of the other agents.

There is also the need for a rule linking the system intentions and the accesses to the databases:

$$assert(inf(A,B,P)) \leftarrow int(A, inf(A,B,P)), isco(P). \tag{8}$$
$$assert(not\ int(A,B,inf(A,B,P))) \leftarrow inf(A,B,P). \tag{9}$$

The first rule defines that, if the system intends to inform the user about some property, then it will access the ISCO database and it will perform an inform action. The second rule means that the inform action will end the intention to perform the inform action!

## 8 Example

Considering the already presented query:

```
Quem salvou crianças?  ‘‘Who saved children?’’
```

The interaction manager receives the query pragmatic interpretation:

```
Q = [child(id=B), toSave(id=C, subject=A, object=B)].
```

After having the sentence rewritten into its semantic representation form, the speech act is recognized:

```
request(user, system, inform(user, system, Q))
```

Using the *request* and the transference of intentions rules the following property is supported:

```
int(system,inform(system, user, Q))
```

Now, using the rules presented in the previous section, the system accesses the ISCO databases and it is able to obtain the final constraints to the discourse referent variables:

```
- $A=_\#(104..109:156..157)$
```

Using the inferred constraints it is possible to obtain the set of solutions to the user query and to answer:

```
m11: Military - rank: colonel; belong: army.
```

## 9 Conclusions and Future Work

A methodology to automatically create legal ontologies was proposed.

The methodology uses syntactical, semantical and pragmatical analysers to obtain sentence representations and to identify entities and entity relations. The obtained ontologies are merged with other externally defined top-level ontologies.

The obtained new ontology is used, with the semantic representation of sentences, to infer class instances and to enrich documents with this semantic information. The inference of the instances associated with each sentence is done via an abductive process – interpretation as abduction.

Ontologies and the inferred instances are represented in the OWL language.

On the other hand, translators from OWL into ISCO/Prolog were developed and a logic programming based interaction manager was developed. The interaction manager uses many important features from its base LP framework: objects, constraints, inheritance.

As future work we need to improve several areas:

– Ontology creation. The ontology was created automatically but it was not possible to identify many relations between the classes. In order to be able to define these relations we intend to extend the statistical analysis of word subcategorisation to take into account semantic information from the dictionary and existent Wordnets.
– OWL translation into ISCO/Prolog. A full translation of the OWL language needs to be implemented and its correction has to be proved.
– Evaluation. The system needs to be fully evaluated and to be tested by users. Moreover it should be applied to other legal documents, such as, legislation.

## References

1. W3C: DAML+OIL – DARPA Agent Markup Language. www.daml.org. (2000)
2. Lassila, O., Swick, R.: Resource Description Framework (RDF) - Model and Syntax Specification. W3C. (1999)
3. Quaresma, P., Rodrigues, I.P.: PGR: Portuguese attorney general's office decisions on the web. In Bartenstein, Geske, Hannebauer, Yoshie, eds.: Web-Knowledge Management and Decision Support. Lecture Notes in Artificial Intelligence LNCS/LNAI, Springer-Verlag (2002)

4. Guangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with dolce. In Gomez-Perez, A., Benjamins, V.R., eds.: Proceedings of the EKAW'2002, Springer-Verlag (2002) 166–181
5. Breuker, J., Winkels, R.: Use and reuse of legal ontologies in knowledge engineering and information management. Journal of Artificial Intelligence and Law (2003) In this issue.
6. Boer, A., Hoekstra, R., Winkels, R., van Engers, T., Willaert, F.: Proposal for a dutch legal xml standard. In: EGOV2002 – Proceedings of the First International Conference on Electronic Government. (2002)
7. Gamallo, P., Agustini, A., Quaresma, P., Lopes, G.: Using semantic word classes in text information retrieval systems. In Pinto, S., ed.: SBIE'2002 – XII Simpósio Brasileiro de Informática na Educação, Workshop de Ontologias, Porto Alegre, Brasil, Unisinos (2002) 593–597 ISBN 85-7431-133-2.
8. Gamallo, P., Agustini, A., Lopes, G.: Using co-composition for acquiring syntactic and semantic subcategorisation. In: ACL-SIGLEX'02, Philadelphy, USA (2002)
9. Lame, G.: Using text analysis techniques to identify legal ontologies' components. In: Workshop on Legal Ontologies of the International Conference on Artificial Intelligence and Law. (2003)
10. Abreu, S.: Isco: A practical language for heterogeneous information system construction. In: Proceedings of INAP'01, Tokyo, Japan, INAP (2001)
11. Alferes, J., Brogi, A., Leite, J., Pereira, L.: Evolving logic programs. In Flesca, S., Greco, S., Leone, N., Ianni, G., eds.: JELIA'02 – Proceedings of the 8th European Conference on Logics and Artificial Intelligence, Springer-Verlag LNCS 2424 (2002) 50–61
12. Damásio, C.: W4 – well-founded semantics for the world wide web. In Boley, H., Grosof, G., Tabet, S., Wagner, G., eds.: Rule Markup Techniques for the Semantic Web, Dagstuhl, Germany (2003)
13. Kamp, H., Reyle, U.: From Discourse to Logic. Kluwer, Dordrecht (1993)
14. Bick, E.: The Parsing System "Palavras". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. Aarhus University Press (2000)
15. Karlsson, F.: Constraint grammar as a framework for parsing running text. In Karlgren, H., ed.: 13th International Conference on Computational Linguistics. Volume 3., Helsinki, Finland (1990) 168–173
16. Gasperin, C., Vieira, R., Goulart, R., Quaresma, P.: Extracting xml syntactic chunks from portuguese corpora. In: TALN'2003 - Workshop on Natural Language Processing of Minority Languages and Small Languages of the Conference on "Traitement Automatique des Langues Naturelles", Batz-sur-Mer, France (2003)
17. Saias, J.: Uma metodologia para a construção automática de ontologias e a sua aplicação em sistemas de recuperação de informação – a methodology for the automatic creation of ontologies and its application in information retrieval systems. Master's thesis, University of Évora, Portugal (2003) In Portuguese.
18. Hobbs, J., Stickel, M., Appelt, D., Martin, P.: Interpretation as abduction. Technical Report SRI Technical Note 499, 333 Ravenswood Ave., Menlo Park, CA 94025 (1990)
19. Alferes, J.J., Pereira, L.M., Przymusinska, H., Przymusinski, T.C., Quaresma, P.: Preliminary exploration on actions as updates. In Meo, M.C., Vilares-Ferro, M., eds.: Procs. of the 1999 Joint Conference on Declarative Programming (AGP'99), L'Aquila, Italy (1999) 259–271
20. Quaresma, P., Lopes, J.G.: Unified logic programming approach to the abduction of plans and intentions in information-seeking dialogues. Journal of Logic Programming **54** (1995)