Languages, Ontologies and automatic grammar generation

Pedro Rangel Henriques

(Mª João Varanda Pereira)

Dep. de Informática
Centro Algoritmi
Universidade do Minho,
Braga-Portugal

UCM
Madrid, 2016-07-12

Motivation

This talk is about:

- Languages
- Formal Languages
- Language Engineering
- Knowledge Representation
- Ontologies
- Ontologies and DSLs

Context

- **Languages** to transmit information and create knowledge;
- **Grammars** to formally specify languages;
- **Ontologies** to formally specify concrete knowledge domains.

Languages and Grammars

From a technical point of view,

A **Language** is a set of sentences.

A **Sentence** is a sequence of Symbols (elements of an Alphabet or Vocabulary)

Languages and Grammars

- Given an Alphabet { a, b, c }

Possible sentences are:

abc, bcca, ca, bac, aabbcc

However to belong to the language (to be *valid*) they must convey with composition Rules

Languages and Grammars

From a social point of view

A Language is essential for communication;

It allows to transmit information from a Sender to a Receiver

- Spoken languages (using sounds)
- Visual languages (using icons/signs)
- Written languages (using characters)

Languages and Grammars

Languages are crucial for the survival of Human Beings

Nowadays, after the invention of computers, new Languages emerged (also crucial for Man-Machine communication)

Languages and Grammars

With a Language we transmit:

- Information to update someone's knowledge base *(Filipe is king of Spain)*
- A query, requiring an answer *(Is Marcelo king of Portugal?)*
- A command requiring an action *(Please give me water)*

Languages and Grammars

- *Computer Languages* can be:
  - General Purpose Programming Languages (GPL)
  - Domain Specific Languages (DSL)

The same basic concept, different application areas => different alphabets, simpler syntax, more expressive semantics

Languages and Grammars

To create a Language it is necessary:

- To choose the **alphabet** (*symbols*) to compose the sentences
- To define the sentences **structure** (*syntax*) – how to combine symbols
- To define the sentences **meaning** (*semantics*) – their effect on the receiver

Languages and Grammars

In the context of Man-Machine interaction, computers need to *recognize* the sentences' *meaning* and *react* accordingly.

To do that automatically (without human intervention) Languages must be *specified formally* !

Languages and Grammars

- To *specify formally* a Language we use:
  - Context Free Grammars (GFG)
  - Translation Grammars (TG)
  - Attribute Grammars (AG)

Many years of research….

Many processing methods (SDT, SemDT)

Powerful Internal Representations (IR)

Languages and Grammars

Context Free Grammars (GFG)

< T, N, S, P (p: X -> X1 Xi Xn) >

Languages and Grammars

Translation Grammars (TG)

< T, N, S, AS, P (p: X -> X1 Xi Xn as) >

Languages and Grammars

Attribute Grammars (AG)

< CFG, A, CR, CC, TR >

Languages and Grammars

Formal Specifications are crucial to build systematically ***efficient*** and ***correct*** Language Processors!

From Grammars we **derive (generate) automatically** Language Processors (compilers)

Languages and Processing (tools)

In this area we like to translate/transform descriptions into another descriptions

allowing the enduser to write closer to his way of thinking (at a more abstract level) hiding details that can be automatically deduced

Languages and Procesing (tools)

- We like to build **Tools**:
  - Compilers/Interpreters
  - Translators/Filters
  - Syntax-directed Editors
  - Validators/Checkers (formal verifications)
  - Code Analyzers
  - Code Profilers
  - Programmer Profilers
  - Model-Model transformers
  - Program Comprehension (concept location)

Languages and Grammars

- To define a Language

1) Choose the Alphabet – the symbols that will be composed to build the sentences

- Not easy
- Close to the domain
- Close to the user customs

Languages and Grammars

- To define a Language

  2) Create the Rules to combine the symbols

- Identify the concepts to be described, as
  - *Variable* – the name of a memory location that holds a value that can change along the execution
  - *Assignment* – the association of a value (defined by an expression) to a variable)
  - *Conditional execution*…

Languages and Grammars

- To define a Language

  2) Create the Rules to combine the symbols

  - Each concept will be represented by a symbol (a non-terminal or terminal)
  - Symbols are composed using the ‘sequence’ operation

Languages and Grammars

To define a Language

3) Define the meaning of each Rule

- Although obvious considering the symbols involved (NTs + Ts)….
- It can be difficult to specify

Languages and Ontologies

- If Languages are setup from symbols that denote concepts

To transmit information and create knowledge

It makes sense to use a knowledge description formalism to understand a domain to help in creating a Language

Languages and Ontologies

This led us to propose the

use of Ontologies to create Languages!

Ontologies

An Ontology describes a knowledge domain, using Concepts and Relations:

$$O = < (C+I), (HR+nHR), P, A >$$

Ontologies

- An Ontology is composed of:
  - set of generic concepts (classes);
  - set of individuals (class instances);
  - set of relations between concepts;
  - set of attributes (name/type-value) characterizing concepts and relations (properties);
  - logical expressions constraining attributes or relations (axioms).

Ontologies

Ontologies are nowadays widely used in the context of Semantic Web and in all applications were it is necessary to search by concepts (looking for semantic content, observing context),

Instead of search by statistics based on the words frequency

Ontologies

Ontologies are used for semantic search,

because they define an alphabet

and an hierarchy of terms and synonymous

Ontologies (examples of SemSearch)

- Query by *Professor* and find *Pedro* that is an instance of that concept;
- Ask something about a *Person* and answer with a *Professor* that is a subclass-of, or with a *Doctor* that is another subclass-of Person;
- Look for something *Good* and find something *Nice* that is a synonymous.

Ontologies

Ontologies provide a conceptual layer, or a more abstract level, over data,

enabling tools to find more accurate answers

Ontologies

Ontologies must be written in formal notation to be stored and handled by computer programs:

- Topic-Maps.
- OWL (Web Ontology Language)
  - is a W3C standard
  - is an extension to RDF / RDF-Schema (Resource Description Framework)

Ontologies

Ontologies can be seen as Graphs and represented by Triples:

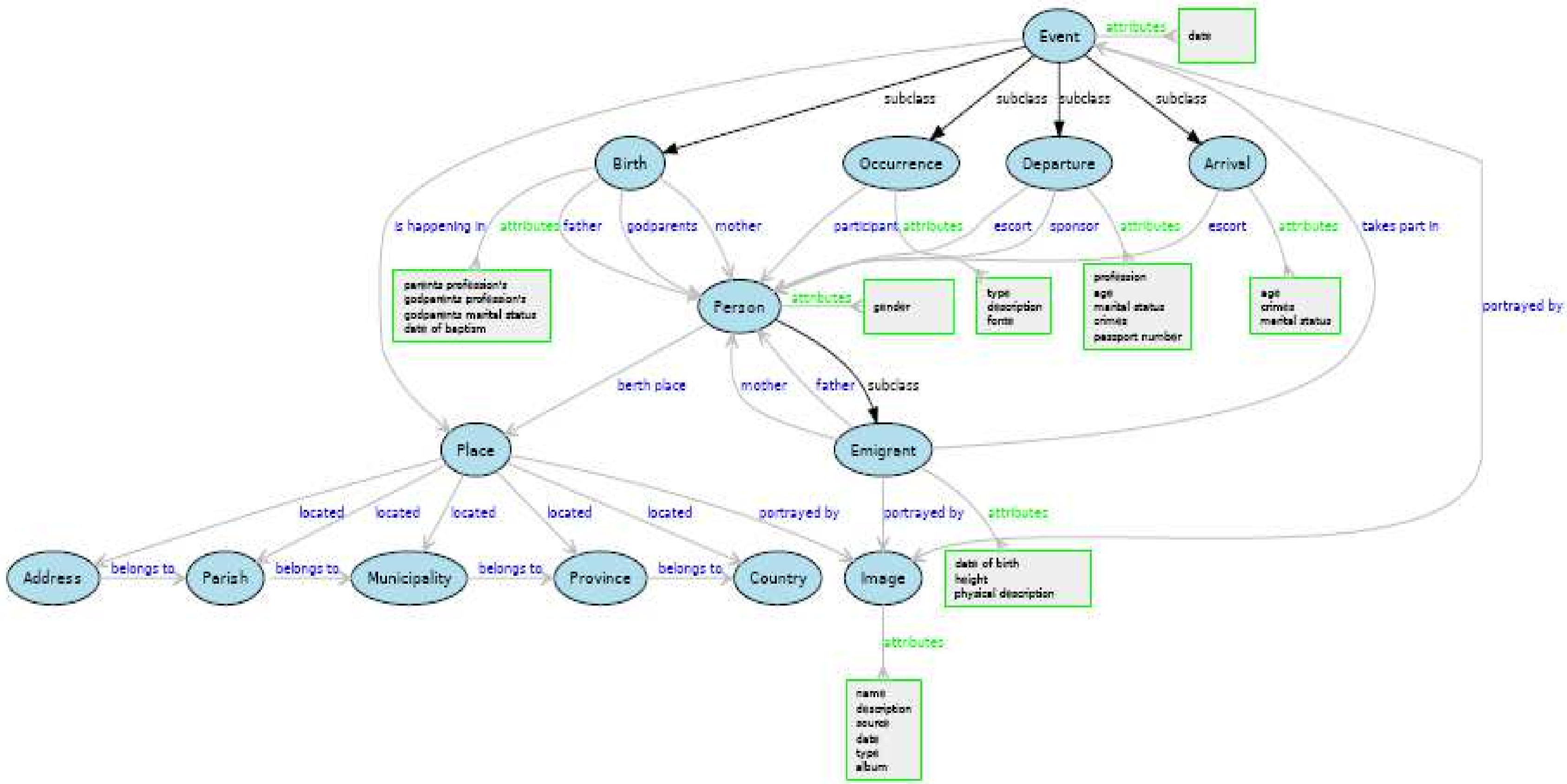
(Subject, Predicate, Object)

Ontologies (case study)

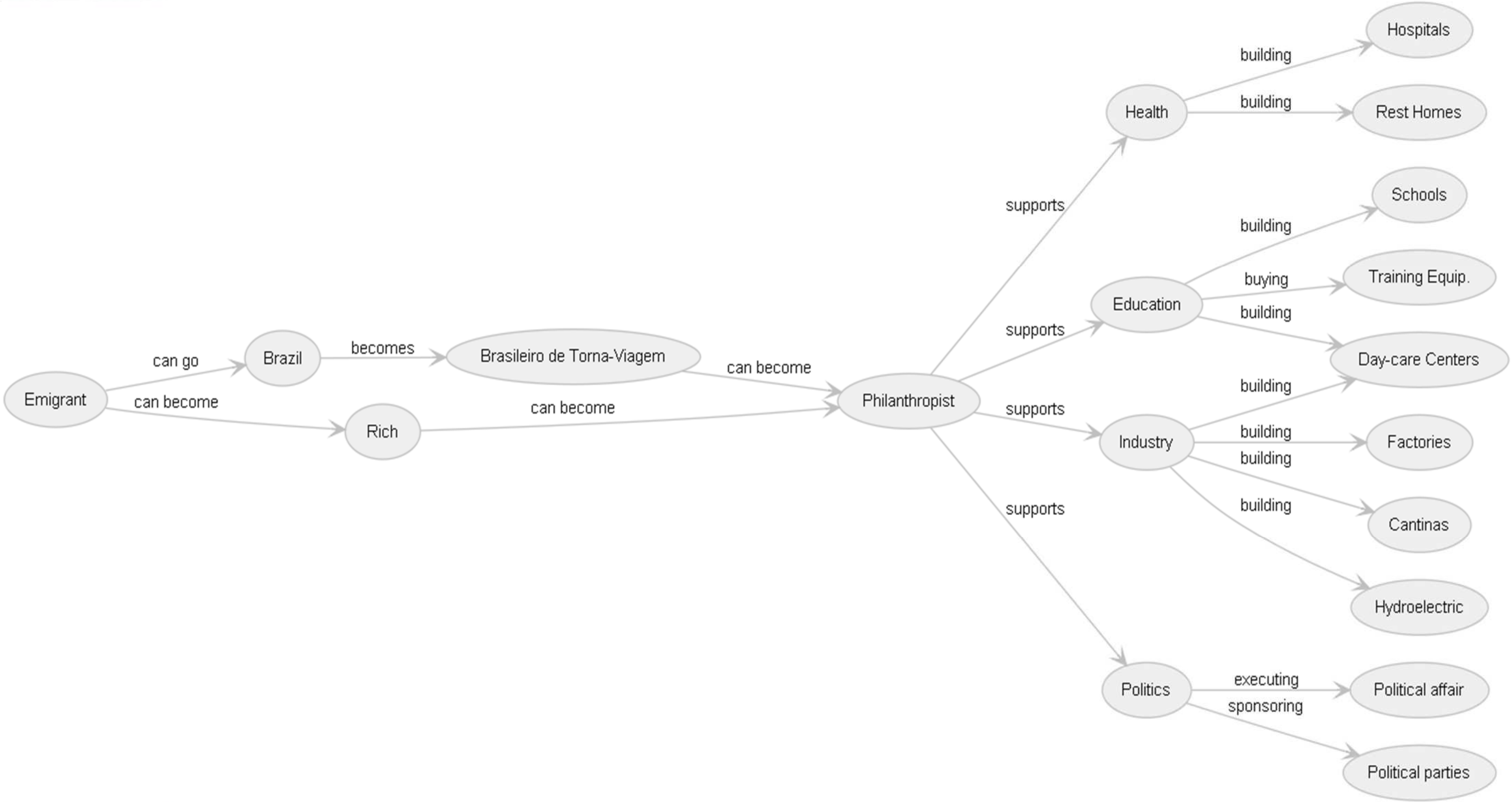
In the context of Virtual Museums, Ontologies provide the basis to markup documents with the appropriated semantic-oriented tags (or annotations)

to extract knowledge or to create relations and links

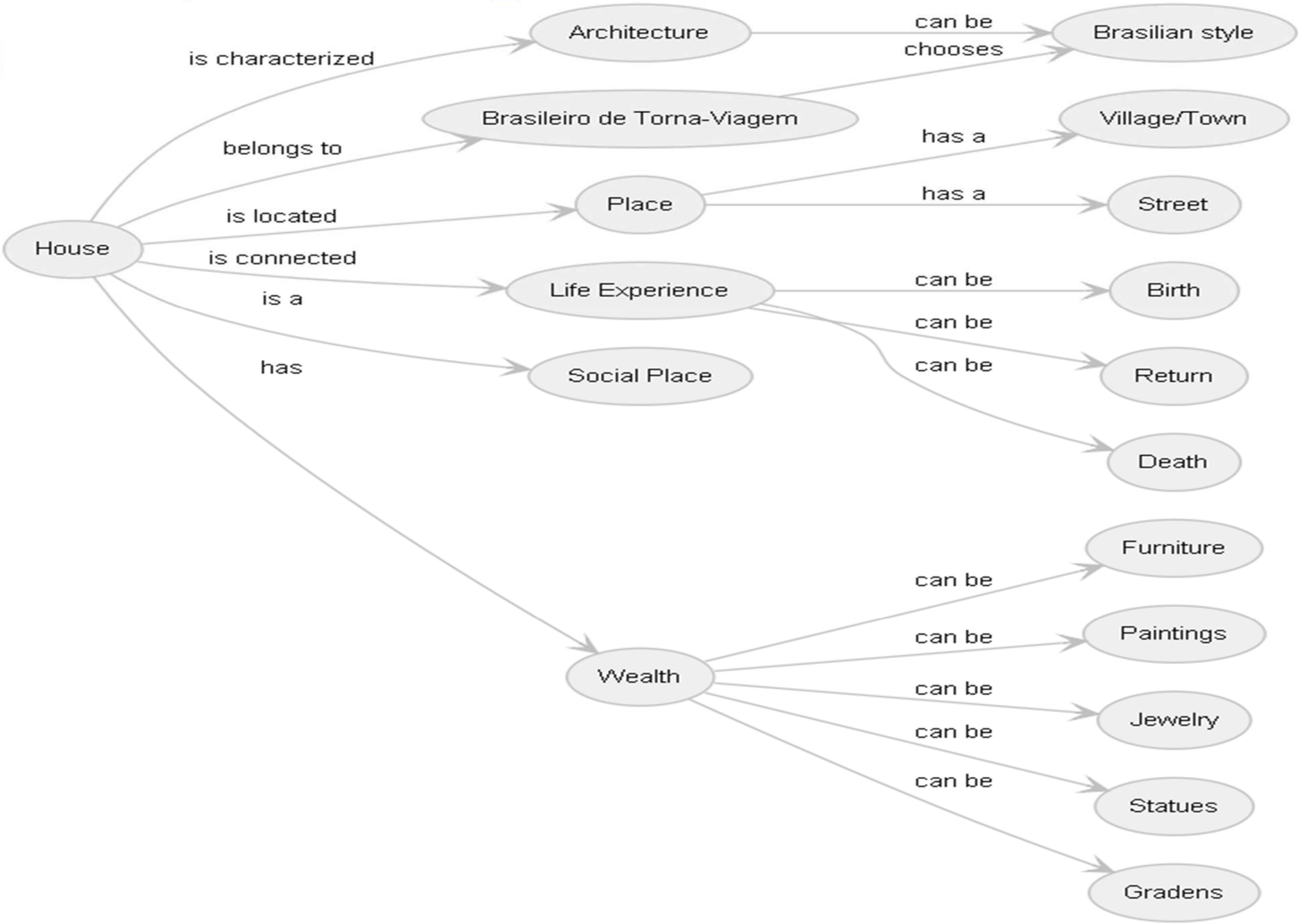
Ontologies (case study)
Example 1: Emigrant Life Events



Ontologies (case study)
Example 2: Emigrant philantropy



Ontologies (case study)
Example 3: Emigrant House



Building rooms with ontologies

Building rooms with ontologies

Building rooms with ontologies

Building rooms with ontologies

Ontologies and Grammars

Has said, we propose to derive a CFG from a given Ontology

based on the simple idea that:

- Concepts are Symbols (Non-Terminal or Terminal)
- Relations and Properties (attributes) give rise to grammar Productions

Ontologies and Grammars

(B, is-a, A) and (C, is-a, A)

||

V

A -> B | C

Ontologies and Grammars

(A, rel1, B) and (A, rel2, C)

||

V

A -> “rel1” B   “rel2” C

Ontologies and Grammars (Ex.)

(Book, has, Author)

(Book, has, Title)

(Author, is-a, Person)

(Reader, is-a, Person)

Book -> “has” Author “has” Title

Person -> Author | Reader

Ontologies and Grammars

(A, att, type)

||

V

A -> “att” type

Ontologies and Grammars (Ex.)

(Author, name, str)

(Author, birth, date)

(Author, rate, int)

Author -> “name” N “birth” B “rate” R

R -> int

B -> date

N -> str

Ontologies and Grammars

Axioms that constraint relations can be translated into:

- Syntactic rules
- Contextual Constraints (require AGs)

Inference rules do not play any role in that process

Ontologies and Grammars

(A, rel1, B [min=1])

(A, rel1, B [max=1])

(A, rel1, B [min=1,max=2])

A -> (“rel1” B)+

A -> (“rel1” B)?

A -> (“rel1” B)(“rel1” B)?

Conclusion

In this talk I intended to review some basic and high-level concepts

like formal languages, grammars and ontologies.

Conclusion

- To conclude I'll present

  a project aimed at automatically

  infer and generate a Grammar for a Domain Specific Language (DSL)

  from a given ontology that describes this specific domain:

  Onto2Gra: "Ontological approach for DSL development".

Onto2Gra

Converting Ontologies to DSL Gramars

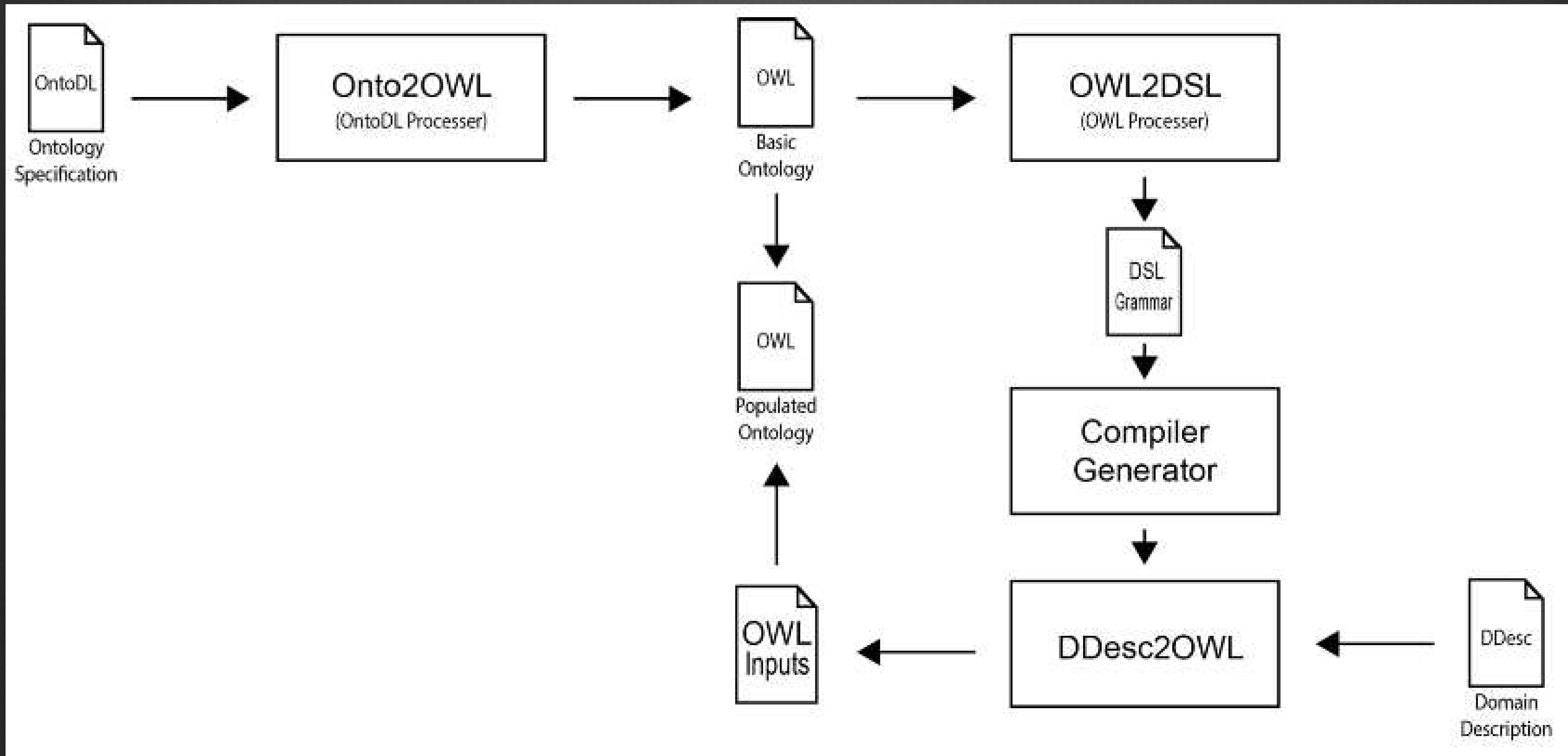
João Fonseca
Maria João Varanda Pereira
Pedro Rangel Henriques

CCTC (UM / IPB)

RESEARCH HYPOTHESIS

Given an abstract ontology, describing a knowledge domain in terms of its concepts and the relations among them, it is possible to derive automatically a grammar to define a DSL for that same domain.

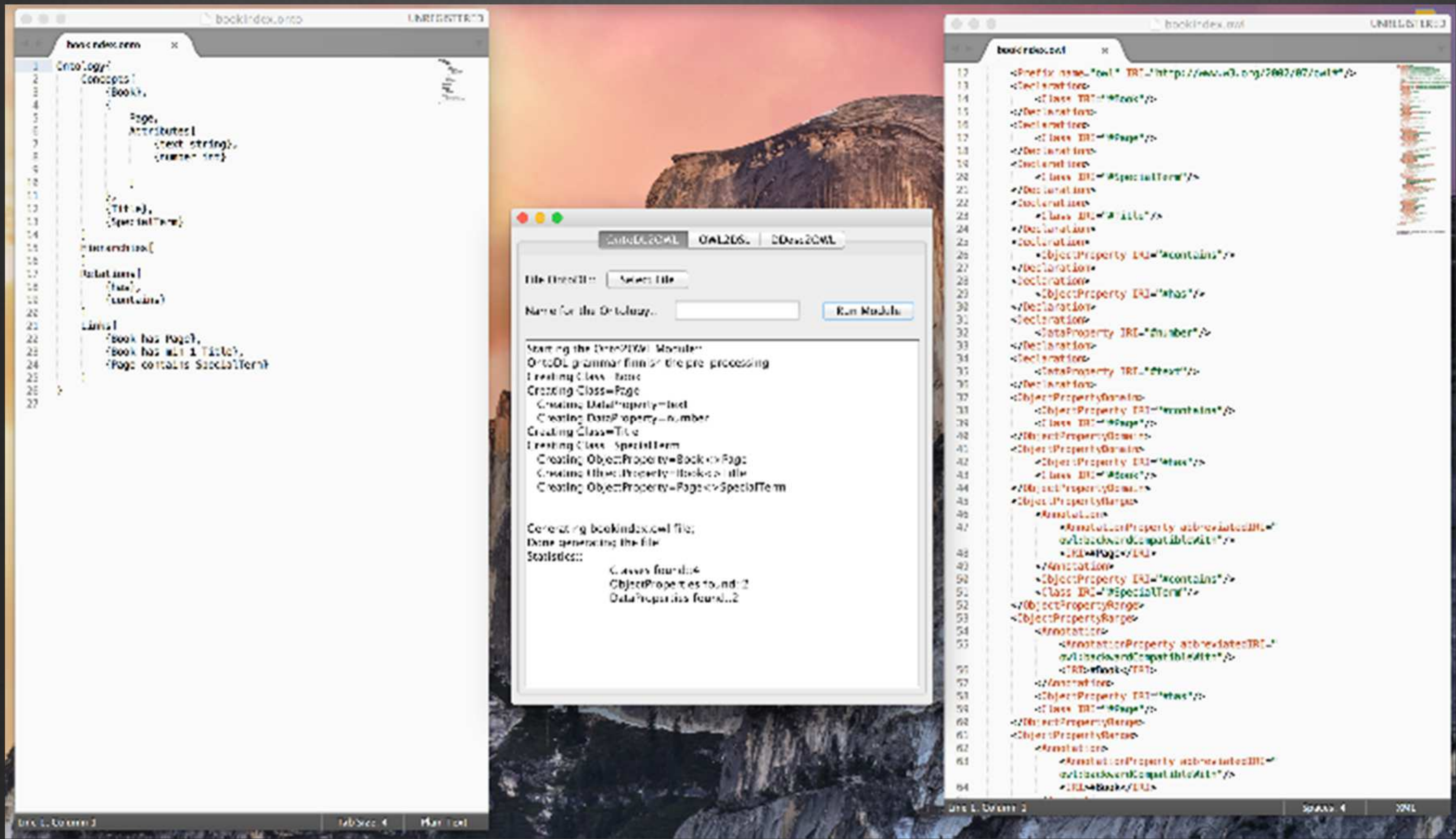
Onto2Gra: the proposal



Onto2OWL

- OntoDL allows
  - simple domain description
  - addition of Cardinality to the Relations between Concepts
- Onto2OWL provides
  - complete OWL generation
  - Including OWL Annotations (they allow a better generation of the grammar).

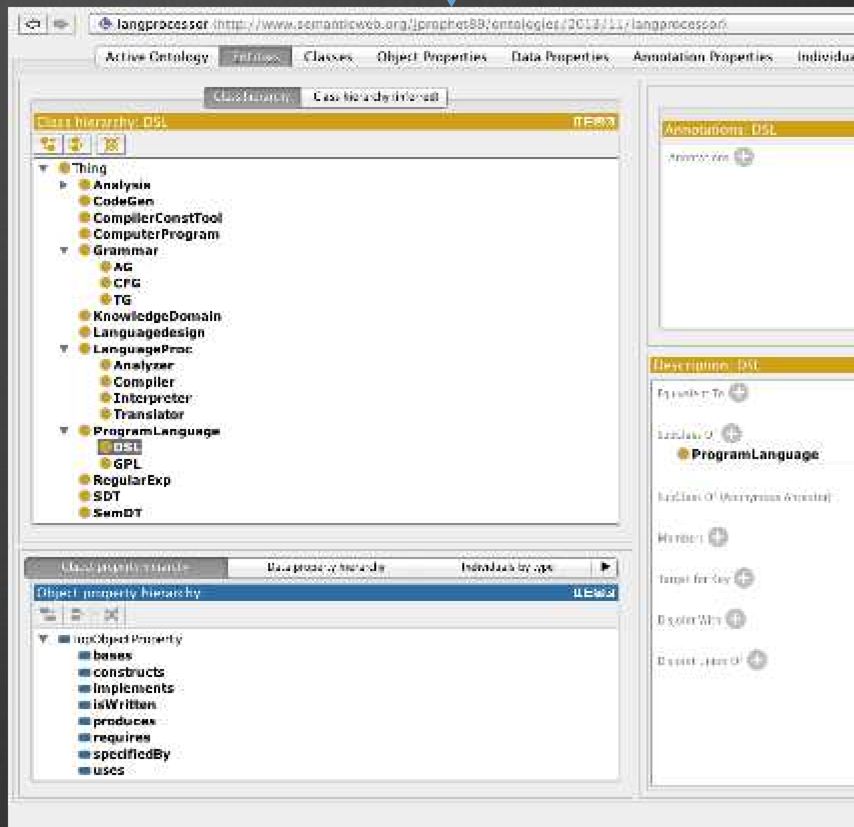
Onto2OWL



Creating Class=Page
Creating ObjectProperty=Book<>Page
Creating ObjectProperty=Page<>SpecialTerm
Statistics::

Onto2OWL

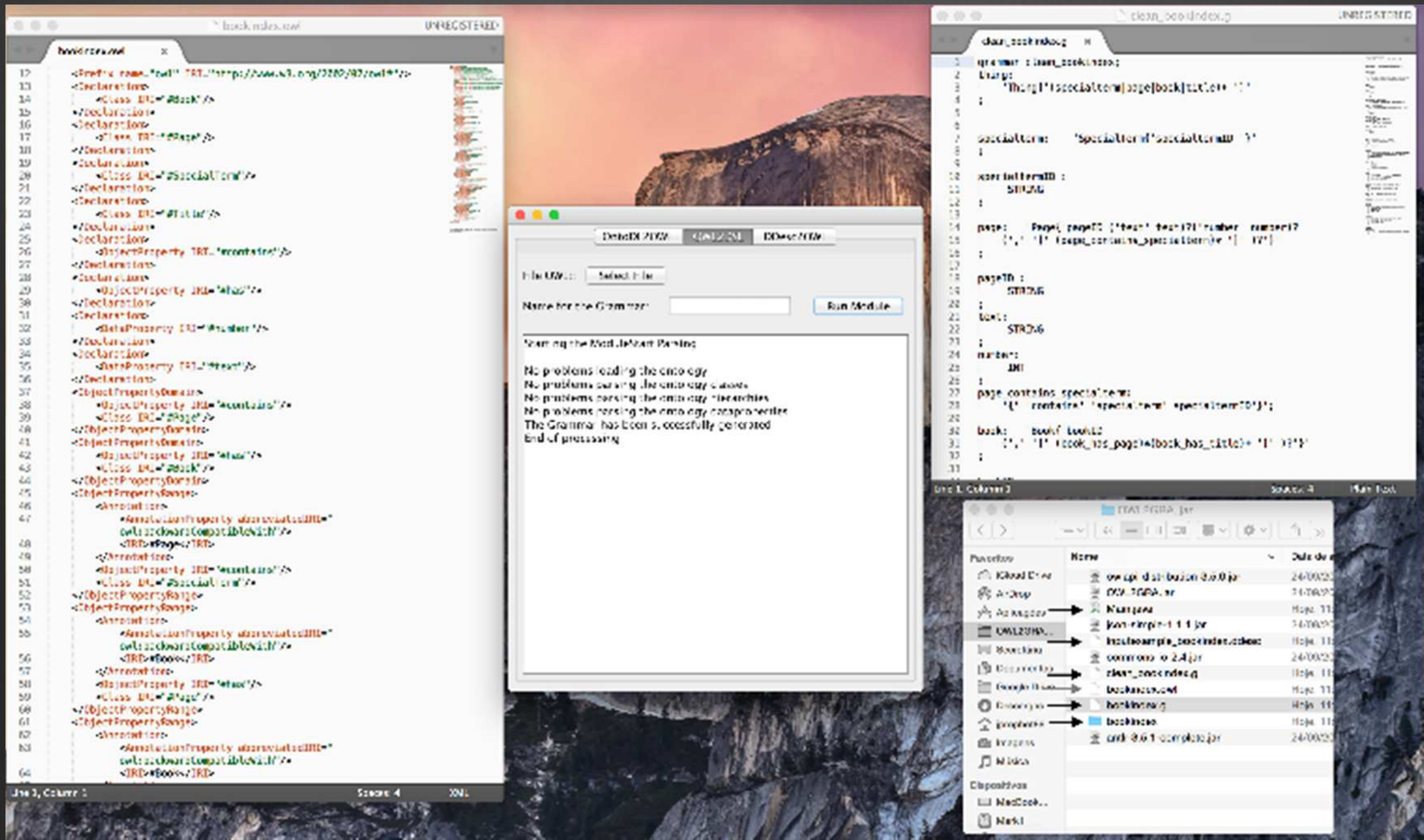
```
Ontology{
    Concepts[
        {KnowledgeDomain},
        {Languagedesign},
        {...},
        {SemDT}
    ]
    Hierarchies[
        {ProgramLanguage > GPL},
        {...},
        {Analysis > SyntacticAnal},
        {Analysis > SemanticAnal}
    ]
    Relations[
        {bases},
        {produces},
        {...},
        {constructs}
    ]
    Links[
        {KnowledgeDomain bases Languagedesign},
        {...},
        {ProgramLanguage specifiedBy min 1 Grammars},
        {SemanticAnal uses min 1 max 2 AG},
        {TG implements max 2 SDT},
    ]
}
```



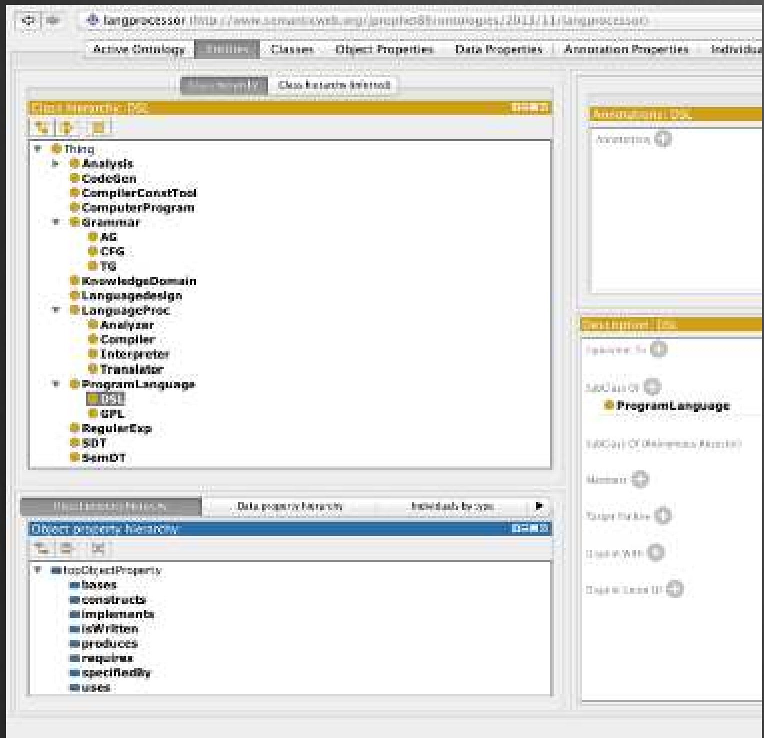
OWL2DSL

- OWL or RDF parsers recognize Concepts, Hierarchies, Data Properties, Object Properties and Annotations
- OWL2DSL generates systematically a grammar applying a simple set of conversation rules
- Four files are generated:
  - one with a Context Free Grammar (only syntactic rules) for readability and for user customization
  - another with an Attribute Grammar (embedding Java code to implement the processor for the last part)
  - The Java Classes
  - The DDesc Template

OWL2DSL



OWL2DSL

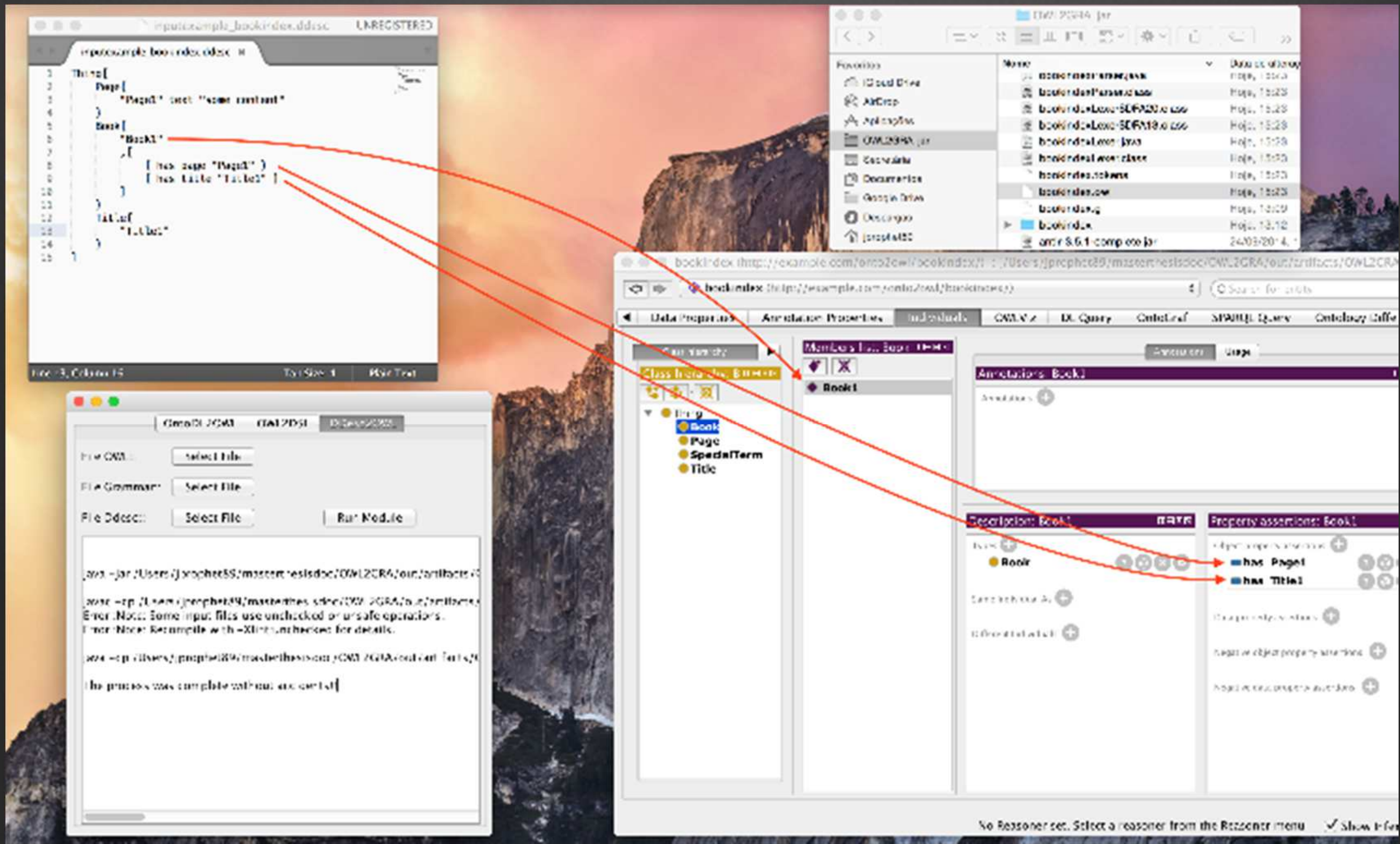



```
grammar langprocessor;
thing:
  'Thing['(knowledgedomain|languagedesign|programlanguage|computerprogram|gram
;


knowledgedomain:  'KnowledgeDomain{'knowledgedomainID
  (',' '[' (knowledgedomain_bases_languagedesign)* ']' )?'}'
;

knowledgedomainID :
   STRING
;
knowledgedomain_bases_languagedesign:
  '{' 'bases' 'languagedesign' languagedesignID'}';

languagedesign:  'Languagedesign{'languagedesignID
  (',' '[' (languagedesign_produces_programlanguage)* ']' )?'}'
;

languagedesignID :
   STRING
;
languagedesign_produces_programlanguage:
  '{' 'produces' 'programlanguage' programlanguageID'}';

programlanguage:  'ProgramLanguage{'programlanguageID
  (',' '[' (gpl
  | dsl)+
 ']' )?
  (',' '[' (programlanguage_specifiedby_grammars)+(programlanguage_specifiedby
;

programlanguageID :
   STRING
;
```

DDesc2OWL

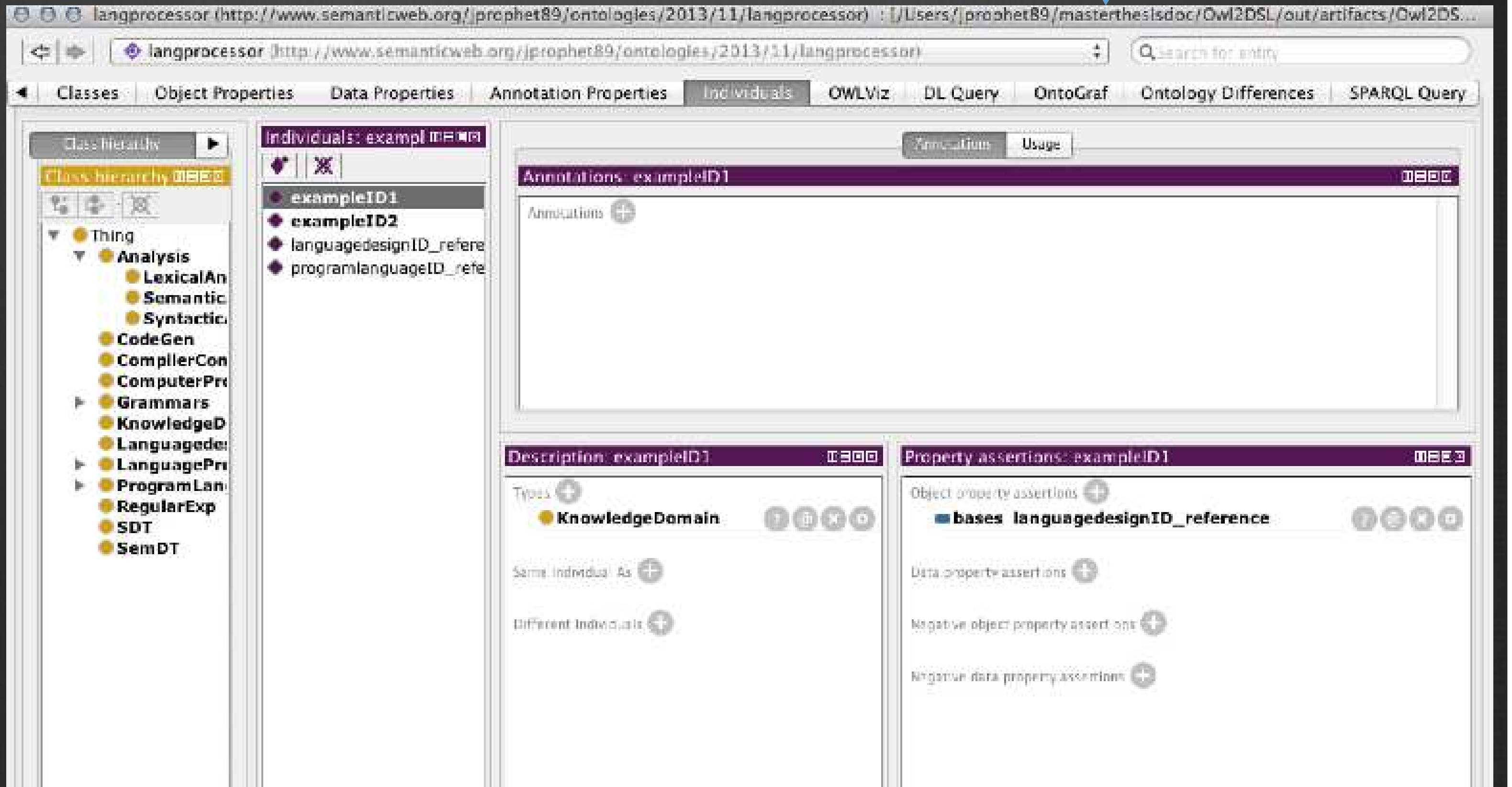
- Easy tool to populate an ontology
- Convert simple Domain Descriptions of Individuals (written in the new DSL defined by the generated grammar) into OWL
- Uses the original ontology (the one used to generate the grammar) and populates it with the inputs from a Ddesc file.

DDesc2OWL



DDesc2OWL

```
Thing[
  KnowledgeDomain{
    "exampleID1"
    ,[   { bases languagedesign "languagedesignID_reference" } ]
  }
  Languagedesign{
    "exampleID2"
    ,[   { produces programlanguage "programlanguageID_reference" } ]
  }
]
```



Conclusion

- The research hypothesis was proved .
- Some improvements were made:
  - Onto2OWL:
    - Supports of a new DSL (OntoDL) for simple Ontology Descriptions
    - Addition of Cardinality to the relations and OWL Annotations for a better grammar generation.
  - OWL2DSL, produces:
    - 2 grammars, the Java Classes to implement DSL processor and the Template for DDesc Module
  - DDesc2OWL
    - Friendly tool that uses the generated DSL processor to populate the original ontology