Extending PROVA for Ontology Definition and Exchange

MFES Cohesive Project: Milestone 4

July 3, 2014

Group D

Let's go Back

Main objective of the project

→ Extend PROVA to support Ontologies
- Study the concept of **Ontology** and the **Web Ontology Language.**
- Study PROVA and its Boilerplates.
- Make a useful translation between OWL and PROVA.
- Construction of a OWL parser to make the previous translation possible.

What is an Ontology?

- → Representation of **knowledge** as a set of concepts of a **specific domain.**
- → Like a **bag of terms and relations.**

Web Ontology Language (OWL)

- Instance of the set theory.
- Classes/Subclasses are just sets.
- *Thing* is the superset representing the universe and *Nothing* the empty set.

Web Ontology Language (OWL)

→ *Classes* can have *attributes*.
→ There are *relations* between the classes.
→ *Individuals* are members of the classes.

Web Ontology Language (OWL)

- Many way of representing the same ontology.
- Different syntaxes to represent an Ontology (RDF only, OWL + RDF).
- The approach of our parser is to parse just a subset of OWL.

PROVA: A new modeling tool

- An innovative **tool** for development of **high assurance systems.**
- **Capable** of **analyzing textual requirements** (with a proper syntax).

PROVA: Structure

- → **Front End** provides a user-friendly way to model the system.
- → **Middle Tier** receives the boilerplates and translate them to be sent to the Back End (Haskell).
- → **Back End** receives the requirements and gives them to a SMT Solver that will return SAT or UNSAT + Model (Haskell).

HASKELL

PROVA: Boilerplates

- "Standardized pieces of text for use as clauses in contracts or as part of a computer program"
- Used to express requirements
- There are three types:
  - Structural
  - Operational
  - Behavioral

OWL parser

- → Written in Haskell to fit into PROVA's back end.
- → OWL Haskell representation was defined.
- → Use of HXT library to help the parsing process.

OWL Haskell :: Ontology

```
data Ontology = Ontology {
                            entities :: [Entity],
                            oProperties :: [Property],
                            dProperties :: [Property],
                            individuals :: [Individual]
                       } deriving (Show, Eq)
```

OWL Parser: Error Handling

OWL parser: error handling

- Easier to debug the parsed ontology
- Attribute of tags not processed by the parser are caught by the error handling system
- Errors are exported to a file

OWL parser: error handling

```
data Error = Error {
                        type_ :: String
                   } deriving (Show,Eq)
```

```
"Invalid attribute type : " ++ attribute ++ ", line : " ++ line
```

- ➔ Error data only contains the type of error
- ➔ Indicate the attribute and the line where the error is caught

OWL parser: error handling

```
getOwlClass :: ArrowXml t => t XmlTree Error
getOwlClass =  atTagE "owl:Class"
            >>>
            proc cl -> do
                    attribute <- (getAttrl>>>getAttrName) -< cl
                    line <- xshow (this >>> changeChildren (take 1)) -< cl
                    returnA -< Error { type = errorOwlClass (qualifiedName
attribute) line }
```

OWL parser: error handling

```
errorOwlClass :: String -> String -> String
errorOwlClass "rdf:about" _ = []
errorOwlClass "rdf:ID" _ = []
errorOwlClass [] _ = []
errorOwlClass attribute line = "Invalid attribute type : " ++
attribute ++ ", line : " ++ (rmvExtraTag line)
```

Valid parsed attribute

Not a valid parsed attribute

OWL parser: error handling output

XML code :

```
<owl:Class href:about="&camera;Damien">
</owl:Class>

<owl:Class rdf:IDe="&camera;Vasco">
</owl:Class>
```

Output :

```
Invalid attribute type_ : href:about, line : <owl:Class href:about="http://www.xfront.com/owl/ontologies/camera/#Damien" />
Invalid attribute type_ : rdf:IDe, line : <owl:Class rdf:IDe="http://www.xfront.com/owl/ontologies/camera/#Vasco"/>
```


From OWL to PROVA

Entities

- All entities are subsets of **Thing:**
  - **Gen “A” idenP “Thing” idenP**
- So the Thing must be declared:
  - **Mult mSome (set ("Thing") [])**

Entities: Example

```
<owl:class rdf:ID=”C”>
    <owl:equivalentClass>
        <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class rdf:about="A"/>
            <owl:Class rdf:about="B"/>
        </owl:intersectionOf>
    </owl:equivalentClass>
</owl:class>
```

Entities: Example

```
Inv "C" (
SCmpF Is (Intersect (set "A" []) (set "B" [])) (set "C"[])
        )
and
Gen “C” idenP “Thing” idenP
```

Entities: Example

```
Inv "C" (
SCmpF Is (Intersect (set "A" []) (set "B" [])) (set "C"[])
        )
and
Gen “C” idenP “Thing” idenP
```

Entities: Example

`Class { name = "A", axioms = [] }`	Gen "A" idenP "Thing" idenP
`Class { name = "A", axioms = [SubClass{ super = Class { name = "B", axioms = []}}] }`	Gen "A" idenP "B" idenP
`Class { name = "A", axioms = [SubClass{ super = Restriction { onProperty = "R", valuesFrom =AllValuesFrom, value = B }] }`	Gen name idenP B ["_R"]
`Class { name = "A", axioms = [DisjointWith{ classes = Class { name = "B", axioms = [] }] }`	Disj ["A","B"]
`Class {name = "A", axioms=[EquivalentTo{ classEqui = Class { name = "B", axioms = []}}] }`	Gen "B" idenP "A" idenP, Gen "A" idenP "B" idenP
`Class {name = "A", axioms=[EquivalentTo{ classEqui = AnonClass { operation = IntersectionOf { setI1 = Class {name= "B", axioms = [] }, setI2 = Class {name = "C", axioms = []} }}] }`	Inv "A" (SCmpF Is (Intersect (set "B" []) (set "C" [])) (set "A" []))
`Class {name = "A", axioms=[EquivalentTo{ classEqui = AnonClass { operation = IntersectionOf { setI1 = Class {name= "B", axioms = [] }, setI2 = Restriction {onProperty = "R", valuesFrom = AllValuesFrom, value = "C"} }}] }`	Inv "A" (SCmpF Is (Intersect (set "R" (["C"])) (set "B" [])) (set "A" []))

Entities: Example

Class {name = "A", axioms=[EquivalentTo{ classEqui = AnonClass { operation = IntersectionOf { setI1 = Class {name= "B", axioms = [] }, setI2 = Restriction {onProperty = "R", valuesFrom = SomeValuesFrom, value = "C"} }}] }	Inv ”A” (MultF mSome (Intersect (set ”R” ([”C”])) (set ”B” [])))
Class {name = "A", axioms=[EquivalentTo{ classEqui = AnonClass { operation = IntersectionOf { setI1 = Class {name= "B", axioms = [] }, setI2 = Restriction {onProperty = "R", valuesFrom = SomeValuesFrom, value = "C"} }}] }	Gen ”A” idenP ”B” idenP, (Inv ”A” (MultF (mJust (v)) (the ”A” [”R”])))

Properties

- **ObjectProperties** are relations, so the boilerplate to be used is the **ASSOC** using the function **has.**
- **DataTypeProperties** are attributes - we should use the **ATTR** boilerplate, but it only supports integer attributes, so it was dealt as a relation.

Properties: Example

```
<owl:ObjectProperty rdf:about="#R">
   <rdfs:domain rdf:resource="A"/>
   <rdfs:range rdf:resource="B"/>
</owl:ObjectProperty>
```

Properties: Example

```
has "A" (From 0) NotFixed "R" "B"
```

Individuals

- The strategy is the same as **Alloy** - we declare the singletons, via **MULT** with multiplicity **one.**
- And we say that the singleton is in the set to which it belongs.

Individuals: Example

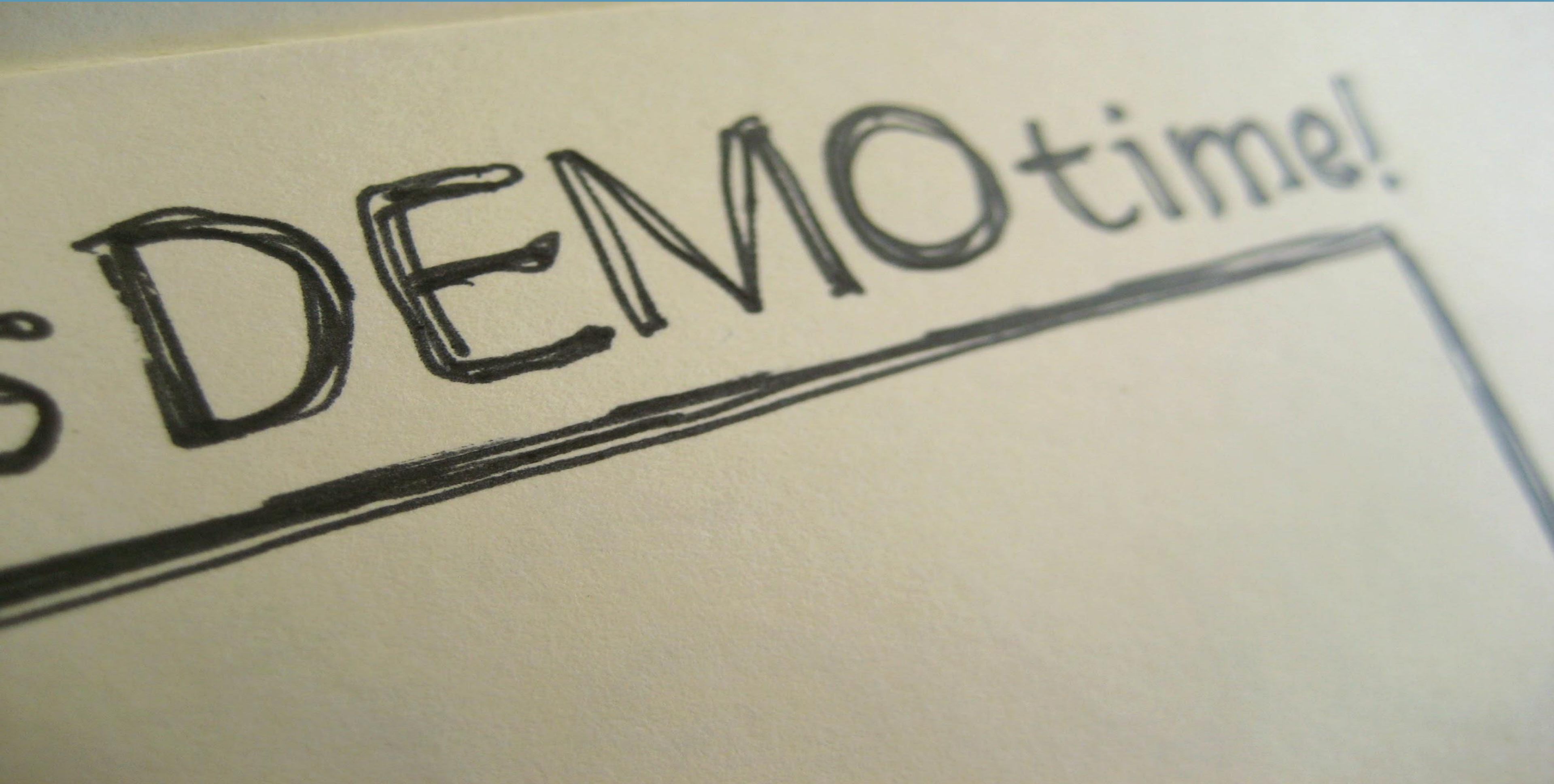
```
<owl:NamedIndividual rdf:about="#c">
   <rdf:type rdf:resource="C"/>
</owl:NamedIndividual>
```

Individuals: Example

```
Mult (mOne) (set ("c") [])
and
Gen "c" idenP "C" idenP
```

Generator of Boilerplates

```
mainBPlateGen :: [ Ontology ] -> [ Boilerplate Name ]
mainBPlateGen  [(Ontology  {  entities  =  e,  oProperties  =  op,
dProperties = dp, individuals = ind})]  =
[Mult mSome (set (Boilerplates.Id.mkName "Thing") [])]
++ (bplateGenEntities e)
++ (bplateGenObjectProperties op)
++ (bplateGenIndividuals ind)
```


DEMO time!

Conclusions
Future Work

Conclusions

- ➔ We learnt about Ontology and Web Ontology Language.
- ➔ We strengthened our knowledge in Haskell
- ➔ We learnt about PROVA's implementation, but it was difficult due to the lack of documentation.
- ➔ However the last point was a little relieved, thanks to our external supervisor.

Future Work

- A wider coverage of OWL by our parser.
- Complete the translation between OWL and PROVA:
  - OneOf Case.
  - And others.

One Last Thing...

Extending PROVA for Ontology Definition and Exchange

pg25300 Damien Vaz   *pg25340 Yoan Ribeiro*

MFES Cohesive Project: Milestone 4