Intelligent Systems ONTOLOGY WEB LANGUAGE AND PROTÉGÉ



Intelligent Systems Lab Universidade do Minho

Tiago Oliveira, Paulo Novais and José Neves

Summary

Ontologies

- What is an Ontology?
- Developing Ontologies Considerations
- Ontology Web Language

Protégé

- Installing Protégé
- Creating a new Ontology
- Entities
- Ontology Definition
- Practical Example Family Ontology
- Bibliography

ONTOLOGIES

What is an Ontology?

A formal explicit description of:

Concepts	Properties	Restrictions
 In a domain of discourse; classes (sometimes called concepts). 	 Of each concept describing various features and attributes of the concept; Slots (sometimes called roles or properties). 	 On slots (facets (sometimes called role restrictions).

Developing Ontologies - Considerations

Why develop an Ontology?

To share common understanding of the structure of information among people or software agents.

□To enable reuse of domain knowledge.

□To make domain assumptions explicit.

To separate domain knowledge from the operational knowledge.

□To analyse domain knowledge.

Developing Ontologies - Considerations

What questions should the Ontology answer?

□Ontology as a Knowledge Base.

What is the underlying domain?

Limiting a domain is complex.

May change during the Ontology development process.

Developing Ontologies - Considerations

Preparing the Domain



Developing Ontologies - Considerations

Standards for Class Development

Classes should be either singular or plural.
(ex: Man is a Persons)

Exception: When a concept only has one form.

□ Class represents the concept, not the word used. □ (ex: Pen and Ballpoint refer to the same concept)

Synonyms and Translations may be added as annotations to classes.

Developing Ontologies - Considerations

Define Hierarchy of Classes

□ Hierarchies are created by **is-a** and **kind-of** relations.

□ (ex: Man is-a Person; Dog kind-of Animal)

□ Subclasses should be at the same level of generalization. □ (ex: Caucasian and Portuguese should not be siblings)

Number of siblings on a class should be between 2 and 12.

Developing Ontologies - Considerations

Define Hierarchy of Classes

□ Hierarchy should be Transitive.

□ (ex: If C is-a B and B is-a A then C is-a A)

□ (ex: Dog is-a Pet and Pet is-a Animal, then Dog is-a Animal)

Multiple Inheritance: a class may be a cubclass of several classes.

□ (ex: Chicken is-a Animal and Chicken is-a Food)

Disjoint Classes: when classes should not share instances.

□ (ex: Man and Woman are commonly Disjoint)

Developing Ontologies - Considerations

Is X a new Class or a Subclass?

Does X have properties which its superclass does not?

Does X have different restrictions from its superclass?

□ Is X involved in relations in which its superclass is not?

□ If yes to any or yes to all, than X is a new class.

Otherwise COMMON SENSE HELPS!

Developing Ontologies - Considerations

Is X a Class or a Property?

□Is X an important term in the Domain?

□It depends on the Domain!

Properties usually represent the most generic concepts?

□Again COMMON SENSE HELPS!

Developing Ontologies - Considerations

Is X a Class or an Instance?

□ It depends on the Granularity of the representation.

□ It depends on the application of the Ontology.

□May X have instances? Then it is a Class.

□ (ex: Maria cannot have instances - Instance)

□ (ex: Car can have instances (BMW) - Class)

Ontology Web Language (OWL)

Standard developed by the World Wide Web Consortium (W3C).

OWL 2

This formalism facilitates machine interpretability and is built upon other technologies such as XML, RDF and RDF-schema.

Ontology Web Language (OWL)

□Sublanguages of OWL 2:

OWL Lite

 Classification hierarchy and simple constraints.

OWLDL

• Description Logics.

 Maximum expressiveness while retaining computational completeness and decidability.

OWL Full

- Different semantics.
- Undecidable.

PROTÉGÉ

Installing Protégé

Go to: http://protege.stanford.edu/

Then go to Downloads

Protégé 4.1 release
 Platform Independent
 Select the OS version
 Include the JVM, if necessary.

- Installing Protégé
 - Launch Protégé.
 - □Select "Create New OWL ontology".
 - Choose the URI.

Choose a directory to store the Ontology.

19

Protégé

Creating a new Ontology

< ontology (http://a.com/ontology) : [C:\Users\Tiago\Dropbox\TOliveira\Apresentações SI\Session 3 \ Session 3 - References\family.swrl.owl] -	
File Edit View Reasoner Tools Refactor Window Help	
Search for entity	
Active Ontology Entities Classes Object Properties Data Properties Annotation Properties Individuals OWLViz DL Query OntoGraf Ontology Differences SPARQL Query	
Ontology header:	
Ontology IRI http://a.com/ontology	
Ontology Version IRI e.g. http://a.com/ontology/1.0.0	
Annotations	
versionInfo	80
1.0	
compent	80
Ontología desenvolvida como turorial pelos alunos do Mestrado Integrado em Engenharia Biomédica	
Ontology imports Ontology Prefixes General class axioms	
Imported ontologies:	
Direct Imports 🕂	
swri (http://www.w3.org/2003/11/swri)	
swria (http://swri.stanford.edu/ontologies/3.3/swria.owl)	
abox (http://swrl.stanford.edu/ontologies/built-ins/3.3/abox.owl)	
tbox (http://swrl.stanford.edu/ontologies/built-ins/3.3/tbox.owl)	
swrib (http://www.w3.org/2003/11/swrib)	
temporal (http://swrl.stantord.edu/ontologies/bullt-ins/3.3/temporal.owl)	
	<u> </u>
To use the reasoner click Reasoner->Start reasoner 🗹 Show I	nferences

ProtégéCreating a new Ontology

O=(C,H,I,R,P,A)

Active Ontology	Entities	Classes	Object Properties	Data Properties	Annotation Properties	Individuals	OWLViz	DL Query	OntoGraf	Ontology Differences	SPARQL Query
0	0	C _c H A	R I	P I		C _i					

Entities

Classes/ Concepts

Restrictions, Enumerations and Partitions

Individuals/Instances

Properties/Relations (Object and Data)

Entities Classes

Description: Thing		
Equivalent To 🛨	Completion (Defined class) Necessary and Sufficient Conditions	
SubClass Of 🛨	Partiality (Primitive class) Necessary Conditions	
SubClass Of (Anonymous Ancestor)	Inheritance from the superclass	
Members 🕂	Instances of the class	
Target for Key 🛨	Properties that uniquely identify the individuals of the class	
Disjoint With 🛨	Classes with which instances cannot be shared	
Disjoint Union Of 🛨	A class C is the union of the class expressions CE_i , $1 \le i \le n$, where all CE are disjoint.	Ξi



If an individual is a member of 'NamedClass' then it must satisfy the conditions. However if some individual satisfies these necessary conditions, we cannot say that it is a member of 'Named Class' (the conditions are not 'sufficient' to be able to say this) - this is indicated by the direction of the arrow.



If an individual is a memeber of 'NamedClass' then it must satisfy the conditions. If some individual satisfies the conditions then the individual must be a member of 'NamedClass' - this is indicated by the double arrow.

Entities

Individuals

Description:	Property assertions:
Types 🕂	Object property assertions 🛨
Classes to which the Individual belongs	
Same Individual As 🛨	Data property assertions 🕂
(ex: Clooney, George Clooney)	
Different Individuals 🕂	Negative object property assertions 🕂
(ex: UCLA, MIT)	
	Negative data property assertions 🕒
	Relations in which the individuals participate.
	with respective concrete values.

Entities

Object Properties

Characteristics: hasNephew	
Functional	
Inverse functional	
Transitive	
Symmetric	
Asymmetric	
Reflexive	
Irreflexive	

Functional: X can only be related with one Y (ex: R = hasMother | X hasMother Y)

Inverse functional: the inverse property is functional

(ex: R = IsMotherOf | its inverse is hasMother)

Transitive: if <u>a R b</u> and <u>b R c</u> then <u>a R c</u>

(ex: R = hasAncestor | X hasAncestor Y and Y hasAncestor Z then X hasAncestor Z)

Entities

Object Properties

Characteristics: hasNephe	ew .	080
Functional		
Inverse functional		
Transitive		
Symmetric		
Asymmetric		
Reflexive		

Symmetric: If <u>a R b</u> then <u>b R a</u>

(ex: R = hasSibling | X hasSibling Y and Y hasSibling X)

Asymmetric: If <u>a R b</u> then not <u>b R a</u>

(ex: R = hasFather | X hasFather Y and not Y hasFather X)

Reflexive: <u>a R a</u> (ex: R = knows | X knows X)

Irreflexive: not <u>a R a</u>

(ex: R = hasBrother | X hasBrother X)

Entities

Object Properties

Description: topObjectProp	erty 🔲 🗏 💷 🖾
Equivalent To 🕂	Property synonyms
SubProperty Of 🕂	Properties of which the current property is a sub-property of
Inverse Of 🛨	Properties representing the inverse relation (ex: hasFather and hasSon)
Domains (intersection) 🛨	The classes having this property
Ranges (intersection) 🕂	The classes targeted by this property
Disjoint With 🕒	Disjoint set of properties
SuperProperty Of (Chain)	Inter-property transitivity (knowledge inference) (ex: X hasParent Y and Y hasParent Z -> X has GrandParent Z)

Entities

Data Properties

Mainly applied to Individuals

Normally it is too general to define a concrete value for a class.

□ (ex: Woman hasAge 25::Integer)

Can only be Functional

□ (ex: Mary hasAge 25::Integer)

Entities

Partitions (Advanced Classes)

Classes based on Partition

Used to refine class descriptions

Restrict the Range

Partitions cover the possible Range of Values

(ex: Child is covered by Daughter and Son)
 A cover axiom is: Child is Daughter or Son
 Daughter and Son are Disjoint

Entities

Restrictions (Advanced Classes)

Classes based on the properties their members may have



Entities

Restrictions (Advanced Classes)

Quantifier Restrictions

Existential (some) or Universal (only)

□ (ex: R **some** Y – means: class of individuals with **at least one** R relationship to Y)

□ (ex: R **only** Y – means: class of individuals with **all** R relationships to Y)

Cardinality Restrictions

Minimum (min), Maximum (max) Exact (exactly)

□ (ex: R min 3 Y – means: class of individuals with minimum of 3 R relationship to Y)

□ (ex: R max 2 Y – means: class of individuals with at most 2 R relationships to Y)

□ (ex: R exactly 1 Y – means: class of individuals with exactly 1 R relationships to Y)

Entities

Restrictions (Advanced Classes)

❑hasValue

□Specifically Valued (value)

□ (ex: R value Y – means: class of individuals with at least one R relationship with the value Y)

Entities

Reasoning

Reasoners that check the consistency of the Ontology
 Protégé uses Fact++ as default
 It helps on inferring new knowledge

□(ex: An individual that belongs to two disjoint classes)

Practical Example – Family Ontology

Terms

Gender		Sister		Child
Parent	Mother	Relative	Female	Daughter
Brother	Father		Person	Aunt
		Son	Uncle	N1 1
Sibli	ng	Niece		Nephew
			Male	

Practical Example – Family Ontology

Hierarchies

Gender Male Female Person Child Daughter Son

Parent Mother Father Sibling Brother Sister Relative Aunt Uncle Nephew Niece

Practical Example – Family Ontology

Adding Classes

	Create Class Filerarchy
	Enter hierarchy
Class hierarchy Class hierarchy (inferred)	Please enter the hierarchy that you want to create. You should use tabs to indent names!
Class hierarchy:	
	Prefix
Thing	Gender
	Person
	Child
	Son
	Parent
↓ ↓ ▼	Father
Remove Class	Sibling
	Sister
	Relative
New Sibling Class	Uncle
U U U U U U U U U U U U U U U U U U U	Nephew
Now Class	NIECE
New Class	
	Go Back Continue Cancel
All at once:	
Tools – Create Class H	lierarchy

Practical Example – Family Ontology

Other Class Operations

Make Siblings Disjoint:
 Select First Sibling and
 Edit – Make primitive siblings disjoint

Make Defined Class from Primitive

- Select Primitive Class and
- Edit Convert to defined class

Practical Example – Family Ontology

Adding Properties

Active Ontology	Entities	Classes	Object Properties	Data Properties	Annotation	
Object property h	ierarchy: to	pObjectPrope	erty			
Ti 🔍 🗙						
e topObje	ctProper	ty				
			$\langle \rangle$			
						Remove Property
						New Sibling Property
						New Sibiling Property
						New Property

Practical Example – Family Ontology

Class Restrictions



Practical Example – Family Ontology

Necessary and Sufficient Conditions

🕶 😑 Person	Primitive Class
🛛 😑 Father	
🔻 😑 Child	
🔤 🖯 🗐 🛑 🛑	
🖯 Son	Description: Father
🔻 🖨 Man	Emiscolart To
🛛 🖨 Brother	Equivalent To
🛛 😑 Nephew	
- Son	Sub Class Of +
🖳 😑 Uncle	hasSon some Son
🔻 😑 Parent	Person
🔤 Mother	



Individuals of Father **are** a Person and are in 'hasSon' relationship with **at least one** individual Son.

```
X implies Y
or
X necessarily holds Y
```

Any individual which **is** a Person and is in 'hasSon' relationship with **at least one** individual Son, **is also** a Father.

X if and only if Y or It suffices Y for X

Practical Example – Family Ontology

Partitions



Practical Example – Family Ontology

Data Properties

	Data property hierarchy: . Ш⊟∎⊠	Annotations Usage	
		Annotations: Age	
	• topDataProperty • Age	Annotations 🕂	
	hasFinishTime hasStartTime		The data type of the
New Data Property			property.
			<u> </u>
		Characteristics	Description: Age
New Sibling Property		Functional	Equivalent To +
Demove Dete Dreperty			SubProperty Of 🛨
Remove Data Property			
			• Person ?@xo
			Ranges 🕂
			integer ?@×0

Practical Example – Family Ontology

Reasoner



Reasoner -> Fact++ Reasoner -> Classify

Inconsistency Problems are marked in red and are placed under the Nothing Class.

Asserted Classes: User-defined classes Inferred Classes: Reasoner-discovered classes and hierarchy.

Practical Example – Family Ontology

Individuals



Practical Example – Family Ontology

Enumerations



Bibliography

- 1. Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. In Development, vol. 32, Nr. 1, pp. 1-25. 2001.
- 1. Ivo Serra and Rosario Girardi. A Process for Extracting Non-Taxonomic Relations of Ontologies from Text. In Intelligent Information Management, vol. 3, Nr. 4, pp. 119-124. July, 2009.
- M. Horridge and H. Knublauch and A. Rector and R. Stevens and C. Wroe. A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools. The University Of Manchester, vol 27, pp. 0-117. 2004.
- 1. OWL Web Ontology Language Overview: http://www.w3.org/TR/owl-features/ . 2004.

Acknowledgements

 Nuno Oliveira, Doctoral Student at the High-Assurance Software Laboratory (HASLab), University of Minho.

In the next session...

 CompGuide – ontology for Computer-Interpretable Guidelines

Intelligent Systems ONTOLOGY WEB LANGUAGE AND PROTÉGÉ



Intelligent Systems Lab Universidade do Minho

Tiago Oliveira, Paulo Novais and José Neves