

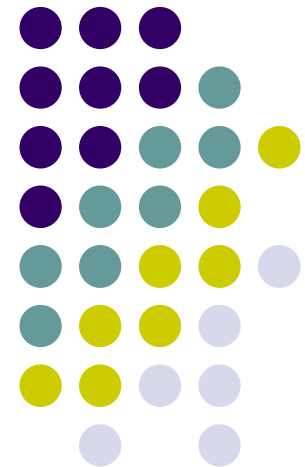
Ontologies (part 3)

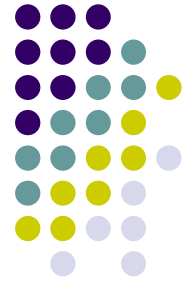
Lina Al-Jadir

UNIGE - SYINF

Cours Bases d'informations

Printemps 2007





Outline

- Introduction
 - Motivations
 - Definition
 - Classification
 - Requirements
- Ontology representations
 - Logic based models:
 - RDF(S)
 - DL
 - OWL
 - Database like models: Kaon
- Conclusion

OWL



- **OWL**: Ontology Web Language
- W3C recommendation, 10 February 2004

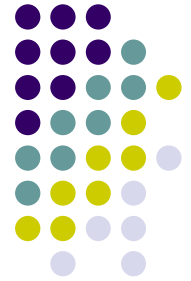
<http://www.w3.org/TR/owl-features/>

<http://www.w3.org/TR/owl-guide/>

OWL



- OWL is the reference ontology language designed for use on the Web
- OWL:
 - = DAML + OIL
(Darpa Agent Markup Language/ Our Ideas of a Language)
 - has 3 roots:
 - Description logics (for formal foundations and reasoning support/ subsumption mechanism) SHIQ DL
 - Frame-based systems (essential modeling primitives)
 - XML based syntax, compliant with RDFS



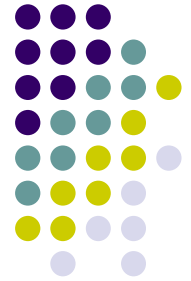
OWL Variants

- *OWL Lite*:
 - limited expressiveness (e.g. cardinality 0 or 1, no union)
- *OWL DL*:
 - greater expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems.
 - OWL DL is so named due to its correspondence with *description logics*
- *OWL Full*:
 - maximum expressiveness and syntactic freedom of RDF with no computational guarantees.



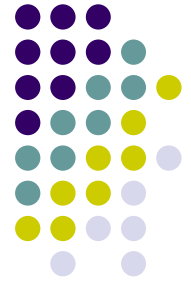
OWL Constructors

Constructor	DL Syntax	Example	Modal Syntax
intersectionOf	$C_1 \sqcap \dots \sqcap C_n$	Human \sqcap Male	$C_1 \wedge \dots \wedge C_n$
unionOf	$C_1 \sqcup \dots \sqcup C_n$	Doctor \sqcup Lawyer	$C_1 \vee \dots \vee C_n$
complementOf	$\neg C$	\neg Male	$\neg C$
oneOf	$\{x_1\} \sqcup \dots \sqcup \{x_n\}$	{john} \sqcup {mary}	$x_1 \vee \dots \vee x_n$
allValuesFrom	$\forall P.C$	\forall hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	\exists hasChild.Lawyer	$\langle P \rangle C$
maxCardinality	$\leq nP$	≤ 1 hasChild	$[P]_{n+1}$
minCardinality	$\geq nP$	≥ 2 hasChild	$\langle P \rangle_n$



OWL Axioms

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human \sqsubseteq Animal \sqcap Biped
equivalentClass	$C_1 \equiv C_2$	Man \equiv Human \sqcap Male
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$ Female
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} \equiv {G_W_Bush}
differentFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg$ {peter}
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter \sqsubseteq hasChild
equivalentProperty	$P_1 \equiv P_2$	cost \equiv price
inverseOf	$P_1 \equiv P_2^-$	hasChild \equiv hasParent ⁻
transitiveProperty	$P^+ \sqsubseteq P$	ancestor ⁺ \sqsubseteq ancestor
functionalProperty	$T \sqsubseteq \leq 1P$	T $\sqsubseteq \leq 1$ hasMother
inverseFunctionalProperty	$T \sqsubseteq \leq 1P^-$	T $\sqsubseteq \leq 1$ hasSSN ⁻



OWL File

```
<?xml version="1.0"?>
```

DTD with entity declarations:

```
<!DOCTYPE rdf:RDF [  
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >  
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >  
>  
>
```

root element rdf:RDF :

```
<rdf:RDF ...>    namespace declarations
```

ontology header:

```
<owl:Ontology> ... </owl:Ontology>
```

ontology classes, properties, individuals:

...

```
</rdf:RDF>
```



OWL Namespaces

- Namespace declarations:

<rdf:RDF

xmlns="http://www.w3.org/TR/2003/PR-owl-guide-20031215/wine#"

xml:base="http://www.w3.org/TR/2003/PR-owl-guide
20031215/wine#"

xmlns:food="http://www.w3.org/TR/2003/PR-owl-guide
20031215/food#"

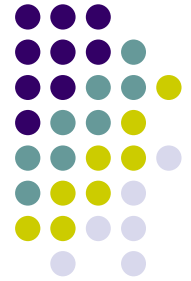
xmlns:owl="http://www.w3.org/2002/07/owl#"

xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

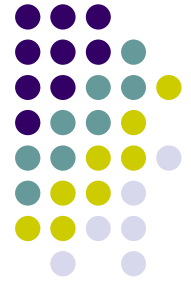
- first 2 declarations identify the (default) namespace associated with this ontology
- 3rd identifies the namespace of the food ontology (prefix food)
- remaining declarations for: **owl**, **rdf**, **rdfs**, **XMLSchema**.



OWL Header

- Ontology header:

```
<owl:Ontology rdf:about="">  
  <rdfs:comment>Exple OWL ontology</rdfs:comment>  
  <owl:priorVersion  
    rdf:resource="http://www.w3.org/TR/2003/  
    CR-owlguide1a/wine"/>  
  <owl:imports  
    rdf:resource="http://www.w3.org/TR/2003/  
    PR-owlguide-1b/food"/>  
  <rdfs:label>Wine Ontology</rdfs:label>  
</owl:Ontology>
```



OWL Classes

- **Classes** correspond to DL concepts

```
<owl:Class rdf:ID="ConsumableThing"/>
```

- rdf:ID defines the class name
- within this document, ConsumableThing can be referenced as rdf:resource="#ConsumableThing"

- **Subclasses:**

```
<owl:Class rdf:ID="Drink">
```

```
<  rdfs:subClassOf rdf:resource="#ConsumableThing"/>
```

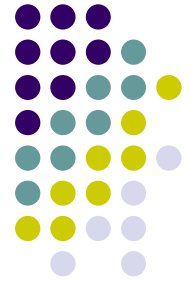
```
</owl:Class>
```

Drink \subseteq ConsumableThing

- Classes and **individuals:**

```
<Drink rdf:ID="water" />
```

Drink(water)



OWL Properties

- 2 types of properties:
 - **datatype properties**: relations between instances of classes and RDF literals and XML Schema datatypes

```
<owl:DatatypeProperty rdf:ID="tannin">  
  <rdfs:domain rdf:resource="#RedWine" />  
  <rdfs:range rdf:datatype="&xsd:string" />  
</owl:DatatypeProperty>
```

- **object properties**: relations between instances of two classes

```
<owl:ObjectProperty rdf:ID="produces">  
  <rdfs:domain rdf:resource="#Winery" />  
  <rdfs:range rdf:resource="#Wine" />  
</owl:ObjectProperty>
```

$\exists \text{produces}.T \subseteq \text{Winery}$ $T \subseteq \forall \text{produces}. \text{Wine}$
--



OWL Properties

- sub-properties:

```
<owl:ObjectProperty rdf:ID="color">  
  <rdfs:domain rdf:resource="#Wine" />  
  <rdfs:range rdf:resource="#WineColor" />  
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />  
</owl:ObjectProperty>
```

...
color \subseteq hasWineDescriptor

- Properties and individuals:

```
<Winery rdf:ID="CC">  
  <name>CortonCharlemagne</name>  
  <produces rdf:resource="#WhiteBurgundy" />  
</Winery>
```

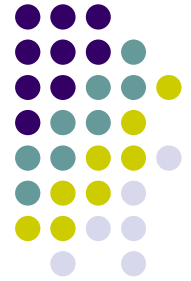
Winery(CC)
name(CC, "CortonCharlemagne")
produces(CC, WhiteBurgundy)

OWL Property Characteristics



- **transitive:**
 - $P(x,y) \text{ and } P(y,z) \Rightarrow P(x,z)$ **locatedIn**
- **symmetric:**
 - $P(x,y) \Leftrightarrow P(y,x)$ **adjacentRegion**
- **inverse:**
 - $P1(x,y) \Leftrightarrow P2(y,x)$ **produces, producedBy**
- **functional:**
 - $P(x,y) \text{ and } P(x,z) \Rightarrow y=z$ **color**
- **inverse functional:**
 - $P(y,x) \text{ and } P(z,x) \Rightarrow y=z$ **produces**

OWL Property Characteristics



- ex:

```
<owl:ObjectProperty rdf:ID="adjacentRegion">  
  <rdf:type rdf:resource="#owl#SymmetricProperty" />  
  <rdfs:domain rdf:resource="#Region" />  
  <rdfs:range rdf:resource="#Region" />  
</owl:ObjectProperty>
```

adjacentRegion \equiv adjacentRegion

...



OWL Property Cardinalities

- It is possible to specify min. and max. cardinalities

```
<owl:ObjectProperty rdf:ID="color">  
  <rdfs:domain rdf:resource="#Wine" />  
  <rdfs:range rdf:resource="#WineColor" />  
  <owl:minCardinality>1</owl:minCardinality>  
  <owl:maxCardinality>1</owl:maxCardinality>  
</owl:ObjectProperty>
```

...
$T \subseteq \geq 1 \text{ color}$
$T \subseteq \leq 1 \text{ color}$



OWL Disjoint Classes

- Disjoint classes:

```
<owl:Class rdf:ID="Food">
```

```
  <rdfs:subClassOf rdf:resource="#ConsumableThing" />
```

```
  <owl:disjointWith rdf:resource="#Drink" />
```

```
</owl:Class>
```

Food \subseteq ConsumableThing Food $\subseteq \neg$ Drink



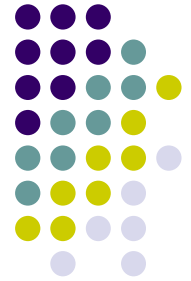
OWL Enumerated Classes

- Enumerated Classes:

```
<owl:Class rdf:ID="WineColor">  
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />  
  <owl:oneOf rdf:parseType="Collection">  
    <owl:Thing rdf:about="#White" />  
    <owl:Thing rdf:about="#Rose" />  
    <owl:Thing rdf:about="#Red" />  
  </owl:oneOf>  
</owl:Class>
```

...

$$\text{WineColor} \equiv \{\text{White}\} \cup \{\text{Rose}\} \cup \{\text{Red}\}$$

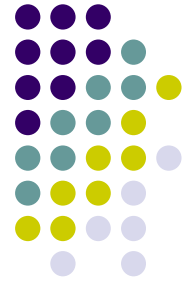


OWL Axioms + Constructors

- A class can be declared as subclass of, or equivalent to, another class B.
 - B is an atomic class
 - B is a complex expression using constructors (*intersection*, *union*, *complement*, property restrictions *allValuesFrom*, *someValuesFrom*, *maxCardinality*, *minCardinality*, *hasValue*)

- `<owl:Class rdf:ID="WhiteBurgundy">`
 - `<owl:equivalentClass>`
 - `<owl:intersectionOf rdf:parseType="Collection">`
 - `<owl:Class rdf:about="#Burgundy" />`
 - `<owl:Class rdf:about="#WhiteWine" />`
 - `</owl:intersectionOf>`
 - `</owl:equivalentClass>`
- `</owl:Class>`

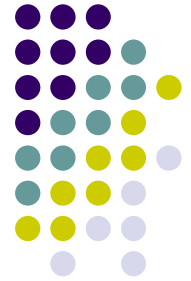
WhiteBurgundy \equiv Burgundy \cap WhiteWine



OWL Axioms + Constructors

- `<owl:Class rdf:ID="Burgundy">`
 - `<owl:equivalentClass>`
 - `<owl:intersectionOf rdf:parseType="Collection">`
 - `<owl:Class rdf:about="#Wine" />`
 - `<owl:Restriction>`
 - `<owl:onProperty rdf:resource="#locatedIn" />`
 - `<owl:hasValue`
 - `rdf:resource="#BourgogneRegion" />`
 - `</owl:Restriction>`
 - `</owl:intersectionOf>`
 - `</owl:equivalentClass>`
 - `<rdfs:subClassOf>`
 - `<owl:Restriction>`
 - `<owl:onProperty rdf:resource="#sugar" />`
 - `<owl:hasValue rdf:resource="#dry" />`
 - `</owl:Restriction>`
 - `</rdfs:subClassOf>`
 - `</owl:Class>`

Burgundy \equiv Wine \cap \exists locatedIn.{BourgogneRegion}
Burgundy \subseteq \exists sugar.{dry}



OWL Axioms + Constructors

- `<owl:Class rdf:ID="Parent">`
 `<owl:intersectionOf rdf:parseType="Collection">`
 `<owl:Class rdf:about="#Person" />`
 `<owl:Restriction>`
 `<owl:onProperty rdf:resource="#hasChild" />`
 `<owl:someValuesFrom rdf:resource="#Person" />`
 `</owl:Restriction>`
 `</owl:intersectionOf>`
 `</owl:Class>`
- $$\text{Parent} \equiv \text{Person} \cap \exists \text{hasChild}.\text{Person}$$

(equivalentClass omitted)

KAON (Karlsruhe Ontology) Approach



- Ontology and semantic web framework allowing for the design and management of ontologies

[A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications,
B. Motik, A. Maedche, R. Volz, ODBASE'02]

KAON



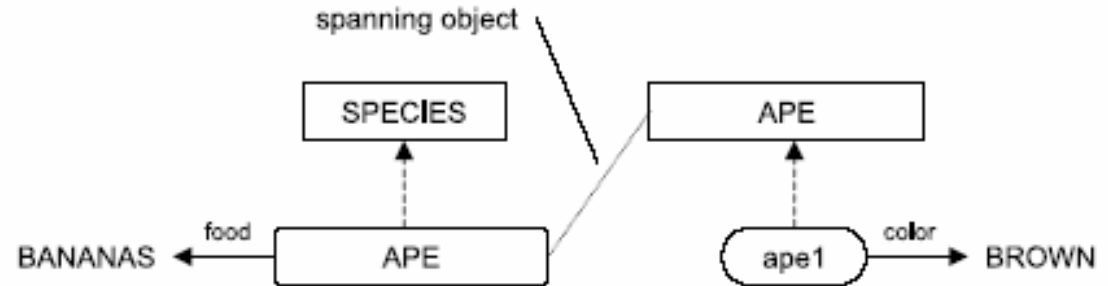
- KAON – ontology management tool suite, contains:
 - Ontology server
 - supports ontology modularization
 - based on relational databases
 - API for ontology manipulation
 - implements ontology evolution
 - OI-Modeler
 - provides multi-user ontology editing
 - Text2Onto – extracting ontologies from texts
 - simplifies ontology development by text mining



KAON

- Supported data sources
 - RDF files
 - engineering server (database based)
- Engineering server (supports ontology engineering)
 - supported databases: Oracle, IBM DB2, MS SQL Server, PostgreSQL

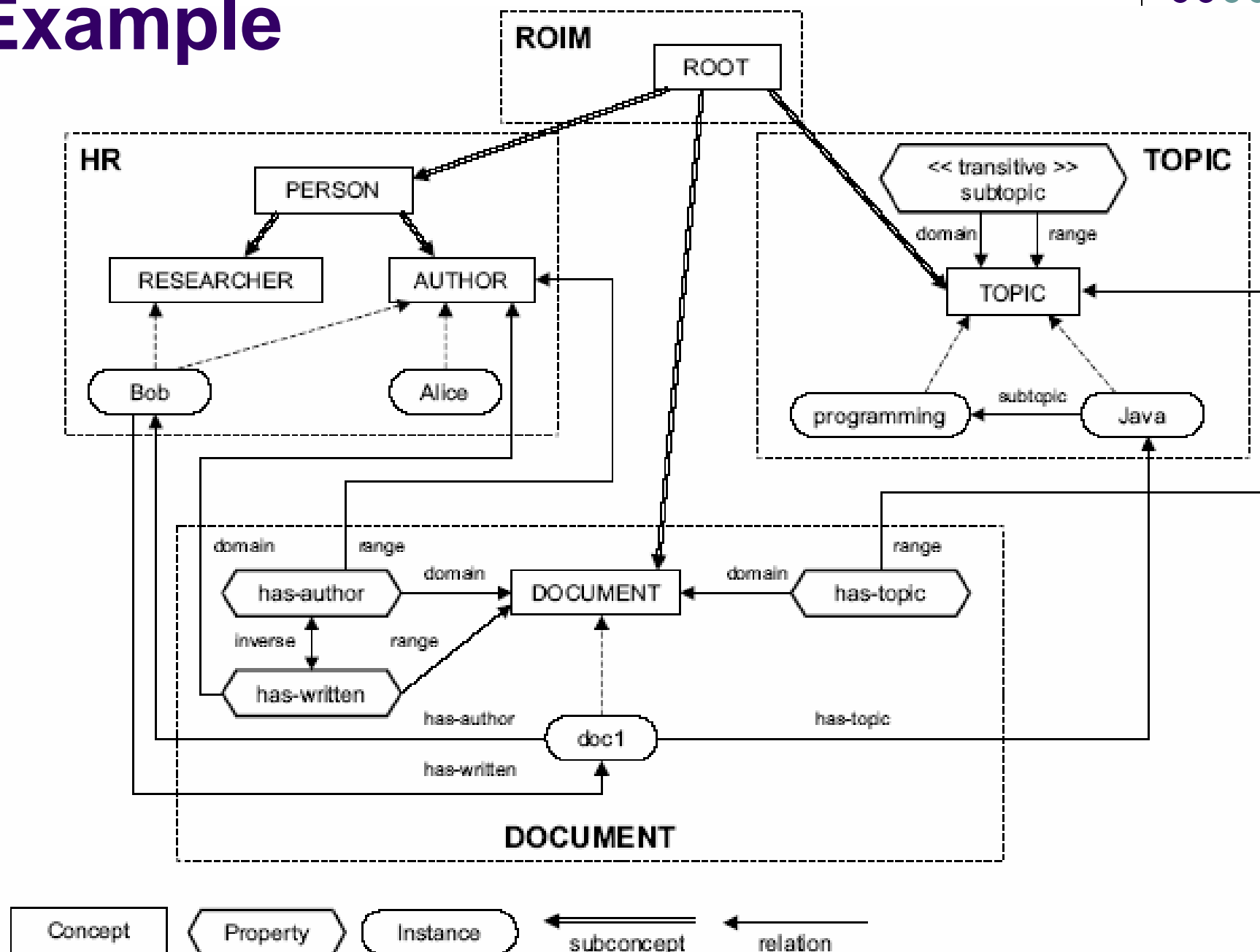
KAON



- Two components:
 - **Ontology structure** (OI-modeler):
 - Definition of concepts.
 - Oriented binary relationships between concepts, and attributes.
 - Relationships may be symmetric, transitive and have an inverse.
 - Minimum and maximum cardinality.
 - Concepts and relationships can be arranged in two distinct generalization hierarchies.
 - **Instance pool**:
 - Concept and relationship instances, and attribute values
 - Spanning objects: real-world entity represented as a concept and an instance



Example



KAON Workbench

File Edit View Procedures

Ol-modeler - file:/C:/Documents%20and%20Settings/christelle/My%20Documents/D/essai.kaon

Zoom

Included Ol-models

- file:/C:/Documents%20and%20Settings/christelle/My%20Documents/D/essai.kaon
- http://kaon.semanticweb.org/2001/11/kaon-ro
- http://kaon.semanticweb.org/2001/11/kaon-le

Search

Execute KAON query

RANGE_CONCEPTS(!#hasWritten!)

Name

- Document

file:/C:/Documents%20and%20Settings/christelle/My%20Documents/D/essai#Sucessful-Author

Superconcepts

Subconcepts

- GoodAuthor

Lexicon

Type	Language	Value
Label	English	GoodAuth...

Properties From Concept

Property...	Minimu...	Maximu...
adress	0	(unlimit...
hasWritten	0	(unlimit...

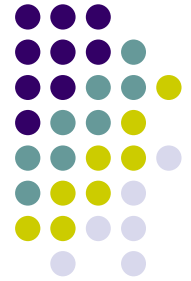
Properties To Concept

Property Name
hasAuthor
http://kaon.semanticweb.org/

Concept Instances

Entity Name	Value
GoodAut...	

Clipboard



Ontology querying in KAON

- Queries on the instances
 - [#Person]
 - [#Person] AND SOME (<#hasWritten>,[#Document])
 - [#Person] AND SOME (<#hasWritten>,SOME (<#hasTopic>,{ !#Java! }))



Ontology querying in KAON

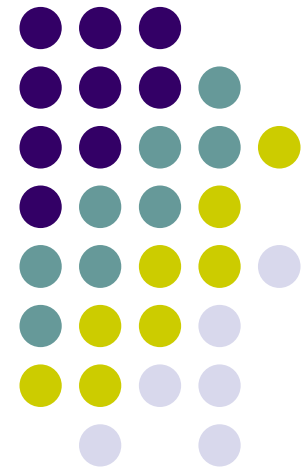
- Queries on the schema
 - Give the domain / range of properties of a concept
 - Give the properties going from (or to) a concept
 - Give all the superconcepts of a concept
 - ...
- Example:
 - SUBCONCEPTS(!#Person!)
 - SUBCONCEPTS^(!#Person!)
 - INSTANCES(!#Person!)
 - PROPERTIES_FROM(!#Author!)
 - RANGE_CONCEPTS(!#hasWritten!)

Ontologies

Introduction

Ontology Representations

Conclusion





Conclusion

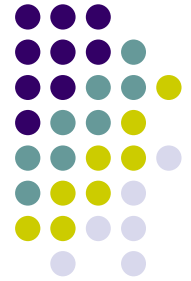
- Ontologies have become a popular research topic in various communities
 - knowledge engineering
 - information retrieval
 - WWW
 - multi-agent systems
 - information integration
 - e-commerce
 - ...
- Lot of application areas



Conclusion

- Use of simple ontologies/taxonomies:
 - Provision of controlled vocabulary
 - Site organization and navigation support
 - Browsing support for users and search engines
 - Improved query evaluation / search support
 - Sense disambiguation for queries

[McGuiness, 2002]



Conclusion

- Use of descriptive ontologies:
 - Consistency checking for knowledge bases
 - Completion of knowledge bases (inferences)
 - Interoperability support (find mappings between ontologies)
 - Structured, comparative, and customized search
 - Exploit generalization/specialization/disjointness information for query optimization, refinement, evaluation, customized presentation and result explanation