Ontologies - II

Web-based Knowledge Representation

Marta Sabou

1

Previous lecture

Semantic Web (SW) - a machine readable web

The SW relies on **metadata**:
* decide on a **model** of the world
* express it in a formal **language**
* describe the content of web pages

Ontology:
* Formal, explicit specification of a shared conceptualization
* a bases to create metadata

2

Outline

1) What is an ontology?
2) Parts of an ontology
3) How to build an ontology?

4) Other issues (characteristics, examples)
5) Employing ontologies
6) Meaning and databases

3

Ontology Elements

Class (concept)
- a name and a set of properties that describe a certain set of individuals in the domain

Property (slot/role)
- assert general facts about the members of a class

Instances
- the members of the sets defined by classes

4

How to build an ontology?

I. - define the skeleton

- 1 - determine domain and scope
- 2 - enumerate important terms
- 3 - decide on classes and their hierarchies
- 4 - decide on properties and their hierarchy

II. - enrich the ontology

- 5 - define classes
- 6 - define properties

5

1: Determine Domain and Scope

Domain: Geography

The science of the earth and life, especially the description of land, sea, air, and the distribution of plant and animal life, including man and his industries with reference to the mutual relations among these diverse elements.

Geography is divided into:

Mathematical Geography
Physical Geography
Biological Geography
Commercial Geography

6

1: Determine Domain and Scope

Application: Route planning agent

The agent finds a route between two specified locations, which can be two cities or two countries. In the later case a route between the capitals of the countries will be returned.

A route consists of more direct connections.

The agent will optimize the route on:

- the number of crossed borders (reduce control)
- number of direct connections. A route can have maximum 12 direct connections.

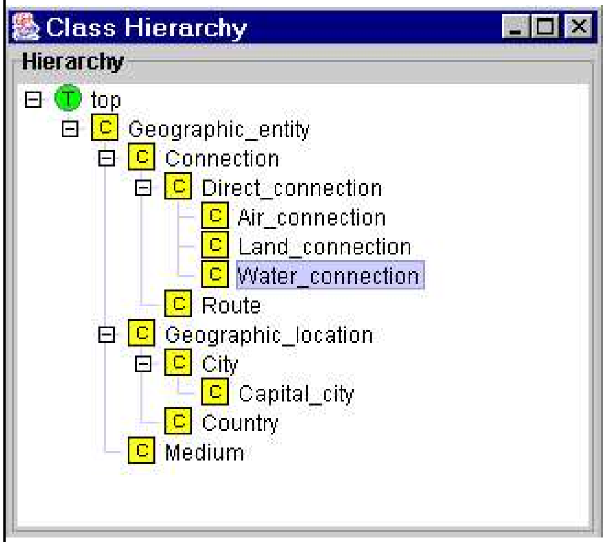
7

2: Enumerate Important Terms

8

3:Decide on Classes and their Hierarchy

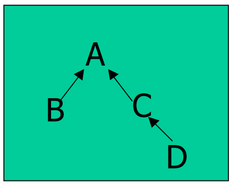
Sub-class hierarchy



C1 subclass of C2
=>
all members of C1 are also members of C2

9

Issues on class hierarchy - I



- do all is-a relations hold?

  $Inst(B) \subset Inst(A)$

- check transitivity

  C Subclass_of(A)
  D Subclass_of(C) } ⇨ D Subclass_of(A)

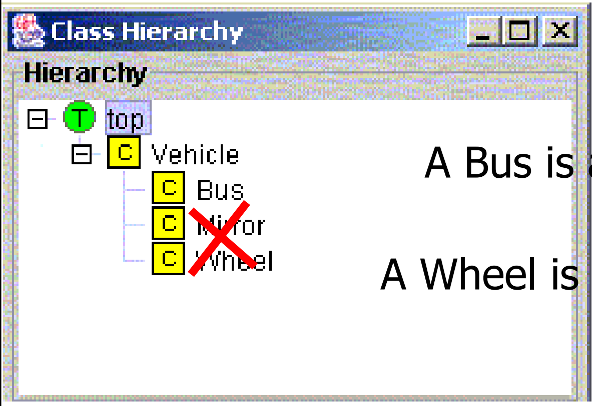
- avoid unexpected cycles

  B Subclass_of(A)
  A Subclass_of(B) } ⇨ A=B

10

Issues on class hierarchy - II

Do not confuse sub-classes with partonomy!



A Bus is a kind of Vehicle.

A Wheel is a part of a Vehicle.

11

4: Decide on properties

Each property has a:
Domain - the class on which the property acts.

Range - the class of values for the property.

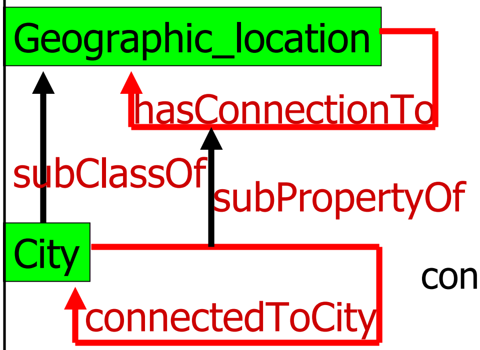
A property:
* describes all subclasses of the domain.
* can take values in any subclass of the range.

12

4: Decide on property hierarchy

P1 sub-property of P2 => if P1(x,y) then P2(x,y)

(The set of pairs related by P1, is a subset of the pairs related by P2.)



City(Paris, Berlin)

connectedToCity(Paris, Berlin)

hasConnectionTo(Paris, Berlin)

13

End of first iteration

Result: a simple ontology containing:

* the name and subclass hierarchy of the main concepts
* the main properties (domain/range) and their hierarchy

Second iteration: enrich the ontology

(5) classes - define classes
* in terms of set-operations
* by property restriction
    * cardinality
    * value

(6) properties - describe their characteristics

14

5:Define Classes -Set operators (I)

Enumeration:

C = {x1, x2, …, x2}

Medium = {Air, Water, Land}

Intersection (logical conjunction):

$C = C1 \cap C2 = \{x | x \subset C1 \text{ and } x \subset C2\}$

Union (logical disjunction):

$C = C1 \cup C2 = \{x | x \subset C1 \text{ or } x \subset C2\}$

Direct_Connection = Land_Connection ∪ Air_Connection ∪ Water_Connection

15

5:Define Classes -Set operators (II)

Complement:

complementOf(C1, C2) => $C1 = \{x | x \not\subset C2\}$

Ex: complementOf (Route, Direct_Connection):

Route = all the Connections which are not direct.

Disjoint:

C1 disjointWith C2 - share no common elements

Ex: disjointWith(Land_connection, Air_connection)

16

5:Define Classes - Property restrictions (I) (cardinality)

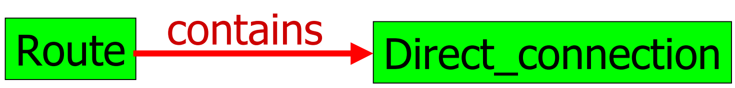
Cardinality restrictions - restricting the number of values a property can have for an instance of a class

* **Exact cardinality**

Cardinality(hasCapital) = 1 :
a country can have exactly one capital

5:Define Classes - Property restrictions (II) (cardinality)

* **Minimal cardinality**



Min_Cardinality(contains) = 1 :
a route contains at least 1 direct connection

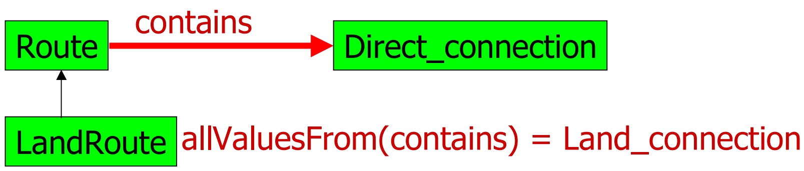
* **Maximal cardinality**

Max_Cardinality(contains) = 12 :
a route contains at most 12 direct connection

5: Define Classes - Property restrictions (I) (value restrictions)

Define classes by restricting the type of values a certain property can have for the members of that class

Universal quantifier (∀) -
all values of the property, if any, should be of the specified type

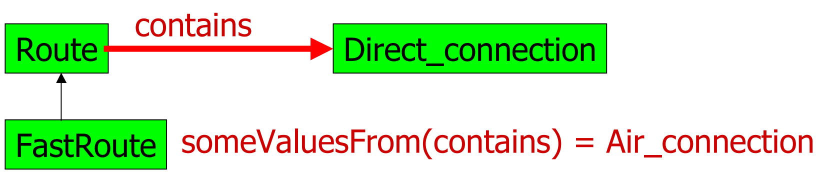


LandRoute - is the route which contains *only* land connections

19

5: Define Classes - Property restrictions (II) (value restrictions)

Existential quantifier (∃) -
the property should have at least one value of the specified type (values of other types are not excluded)

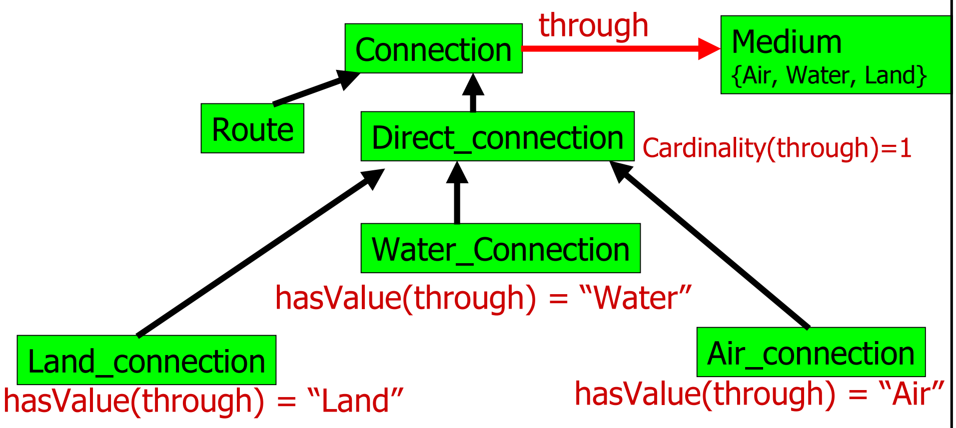


FastRoute - is a route that contains at least one Air_connection, but can contain connections of other types too.

20

5:Define Classes - Property restrictions (III) (value restrictions)

Named value - the property should have at least the specified value (other values are not excluded)

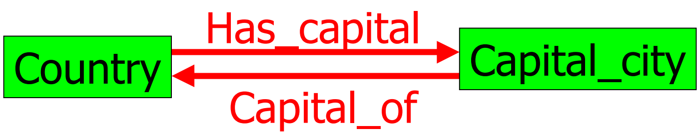


21

6: Property Characteristics (I)

* *Inverse* property (redundant, but explicit)

P1 inverse of P2: P1(x,y) => P2(y,x)



* *Symmetric* property

P symmetric: P(x,y) => P(y,x)

borders_with(A, B) ⟺ borders_with(B,A)

22

6: Property Characteristics (II)

* *Transitive* property

P transitive: P(x,y) and P(y,z) => P(x,z)

connectedToCity(Amsterdam, Utrecht)
connectedToCity(Utrecht, Maastricht)

connectedToCity(Amsterdam, Maastricht)

* *Functional* property (*unique*)

P functional: P(x,y) and P(x,z) => y = z

(P assigns at most a single value to any member of its domain)

Ex: hasCapital assigns a single city as capital for any country

23

Adding instances

Country(Netherlands)
Capital_city(Amsterdam)
Capital_city(The Hague)
City(Utrecht)

hasCapital(Netherlans, Amsterdam)
hasCapital(Netherlans, The Hague) ???
= hasCapital is unique and has cardinality 1

hasCapital(Netherlans, Utrecht) ???
Utrecht is just a City, not a capital

24

Remarks- building an ontology

- influenced by the type of application
- iterative
- there is no single way to model a domain

More: McGuinness, Noy --
"Ontology development 101: A Guide to Creating Your First Ontology"

25

Outline

1) What is an ontology?
2) Parts of an ontology
3) How to build an ontology?
4) Other issues (characteristics, examples)
5) Employing ontologies
6) Meaning and databases

26

Amount of meaning

- attribute of the ontology

Amount of meaning:

- the more meaning the lower ambiguity

•**Light-weight** ontologies

-taxonomy - only class hierarchy

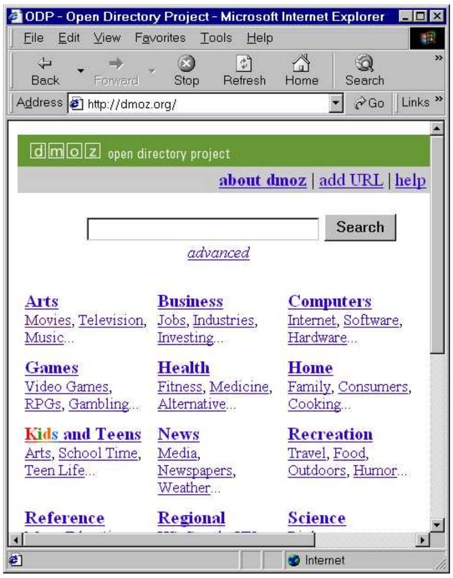
YAHOO! ODP(www.dmoz.org) } *web catalogues*

•**Heavy-weight** ontologies

Ex: Cyc ontology - common knowledge of the world

27

Amount of meaning - light weight



- taxonomies
- no pure subclass relationships

28

Amount of meaning - heavy-weight

Example: CYC (Lenat & Guha)

#$Skin

A (piece of) skin serves as outer protective and tactile sensory covering for (part of) an animal's body. This is the collection of all pieces of skin. Some examples include TheGoldenFleece (an entire skin) and YulBrynnersScalp (a small portion of his skin).
isa: #$AnimalBodyPartType
genls: #$AnimalBodyPart #$SheetOfSomeStuff #$VibrationThroughAMediumSensor #$TactileSensor #$BiologicalLivingObject #$SolidTangibleThing
some subsets: (4 unpublished subsets)

- *Note*: some roles (*of*, *through*, …) implicitly encoded into the naming of entries: but likely not accessible as such

Degree of formality

- attribute of the ontology representation language

Highly informal - expressed in loose natural language
Ex: glossaries, WordNet

Informal - expressed in structured natural language

Semi-formal -expressed in an artificially defined language

Rigorously formal - terms with formal semantics, theorems, proofs
Ex: Cyc

Outline

1) What is an ontology?

2) Parts of an ontology

3) How to build an ontology?

4) Other issues (characteristics, examples)

5) Employing ontologies

6) Meaning and databases

Employing Ontologies

There are three scenarios:

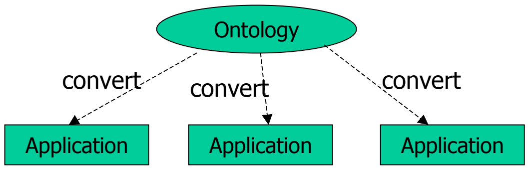
differ by the solved problem and
the role of the ontology

- neutral authoring
- common access to information
- indexing

M.Uschold, R. Jasper, *"A framework for understanding and classifying ontology applications"*

Neutral Authoring

"An information artifact is authored in a single language, and is converted into a different form for use in multiple target systems "



-decreased cost of reuse

-portability of knowledge

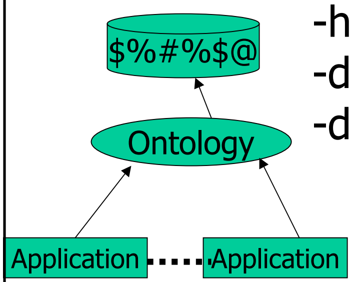
-application maintainability

33

Common Access to Information

Context: information is required by one or more persons or computer applications, but is expressed in an unfamiliar vocabulary.

Role of ontology: helps to render the information intelligible by providing a shared understanding of terms/ mapping between sets of terms.



-human communication
-data access via shared ontology
-data access via mapped ontology

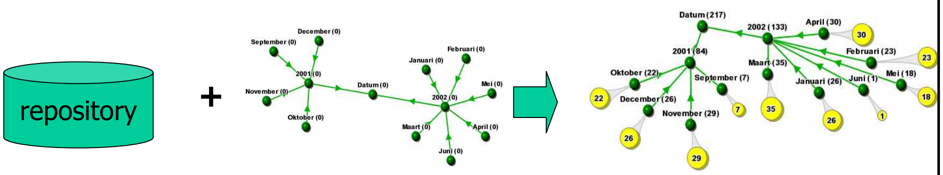
-interoperability
-knowledge reuse

34

Indexing

Context: "Location of artifacts in some repository"

"The ontology is used to index information artifacts"



Used by: Yahoo!, Google etc.

35

Application Fields

- knowledge engineering
- natural language processing
- cooperative information systems
- intelligent information integration
- knowledge management
- electronic commerce
- information retrieval
- multi-agent systems

36

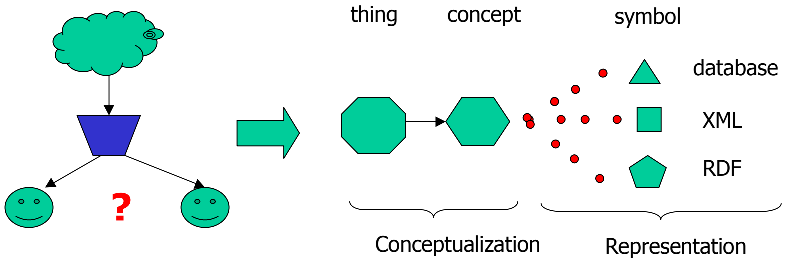
Outline

1) What is an ontology?
2) Parts of an ontology
3) How to build an ontology?

4) Other issues (characteristics, examples)
5) Employing ontologies
6) Meaning and databases

37

Expressing meaning



For any ontology representation language:
- data model
- the expressiveness
- querying

First: DATABASES

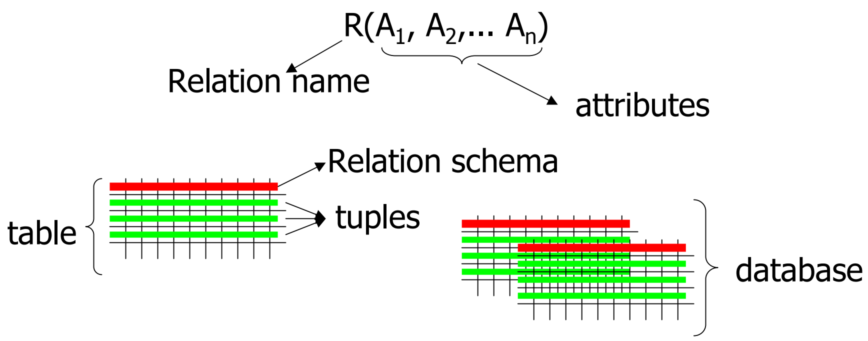
38

Relational Data-Model

- introduced by T. Codd '70
- most used
- simple
- mathematically founded
  - set theory
  - first order predicate logic

39

The model



$S=\{R_1, R_2,\ldots R_m\}$ ← *the database schema*

The metadata - describes *the structure* of the repository

40

Geography domain in database

What tables would you have for this ontology?

Ex: Country(Name, Borders_with, Has_capital)

How can I say:

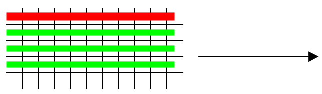
Country is a subclass of *Geographic_entity*?

Ex: Country(Name, Borders_with, Has_capital, Subclass_of)???

Does a machine really *understand* this?

41

What's it like to be a computer? *(Symbol version)*



#%$^%	##	^&
	%^	$$
	#@	*&
	*&^	&^

Machine still does not understand the words, but it knows how they are related.

42

Querying a relational database

Operations:

- selection
- projection

} Relational model

- union
- difference
- intersection

} Set theory

The result of each operation is a relation, on which new operations can be applied.

43

Database querying

Selection:

$\sigma_{<\text{selection condition}>}(R)$ (selects certain rows)

Ex: $\sigma_{<\text{Has\_capital=Helsinki}>}(\text{Country})$

The resulting relation has the same attributes.

Projection:

$\Pi_{<\text{attribute list}>}(R)$ (selects specified columns)

Ex. $\Pi_{<\text{Name, Has\_capital}>}(\text{Country})$

The resulting relation has the *selected* attributes only.

44

Database querying - *Set Theory Operations*

- binary
- complement the other operations

Union: $R \cup S$

Intersection: $R \cap S$

Difference: $R \setminus S$

Conclusion:
The operations are biased by the data model.

45

Remark

The relational model expresses limited ***meaning*** (relationship between entities)

It is suited for applications that need ***the structure*** of the information.

Ontology	Database schema
Domain theory (specialization, constraints etc.)	Structure of a data repository
Shared	Locally used

46

Take home message

Ontologies are shared domain models.

Ontologies foster communication.

You have learned an informal ontology about ontologies.

Next Lessons:
- how to formally represent ontologies

47

Assignment

Task: build an ontology
* domain:
* scope: defined by you
* first iteration:
    * a few classes + hierarchy
    * a few properties(domain/range)
* second iteration:
    * define a class
    * add a special property
* add some instances

Hand in a Report (1/2 pages.)

48