2. Knowledge Representation and Communication

Part 1: Knowledge Representation

Javier Vázquez-Salceda
SMA-UPC

Communication Agents...

- Mutual understanding:
  - Translation between representation languages
  - Share the language’s semantic content

- Three components in communication:
  - Interaction protocol
    - How are conversations/dialogues structured?
  - Communication Language
    - What does each message means?
  - Transport protocol
    - How messages are actually sent and received by agents?
Communication and Knowledge Level

- Agents can be considered as (virtual) Knowledge Bases
- 3 representation levels
  - A language/formalism to represent domain knowledge
    - Ontology
  - A language to express propositions (to exchange knowledge)
    - Content language (for messages)
  - A language to express attitudes for those propositions
    - Agent Communication Language (for languages)

Knowledge Representation

- Motivation
- Ontology Design
- Languages for Knowledge exchange
Ontologies

- Ontology science aims to study the categories that exist in a given domain.
- The result of this study is an ontology.
  - A catalogue of the different kinds of objects that we assume as existing in a given domain $D$, from the perspective of someone that uses a language $L$ in order to talk about $D$.
- Elements in ontologies represent predicates, constants, concepts and relationships.
- An ontology can be seen as the vocabulary that agents need to use in order to talk about a given domain.

Motivation

- To allow sharing an interpretation of information structure between people/agents
  - By creating an ontology about a domain, agents can understand each other (unambiguously) and know what the other means with each message.
- To allow knowledge reuse
  - Create a domain description which can be used by other applications which should use/share knowledge about that domain.
- To make explicit the interpretations about the domain
  - Interpretations about concepts, predicates… can be compared. If conflicts arise, a common interpretation can be agreed upon.
Ontologies

Motivation

- Ontologies divide domain knowledge from operational knowledge
  - Allows to independently develop the techniques and algorithms to solve a problem from the concrete knowledge about the problem

- They allow analysis over domain knowledge
  - Once we have a knowledge specification, it can be analysed by means of formal methods (correctness, completeness …)

Design and development

- Creating an ontology requires
  - To define the classes in the domain
  - To organize the classes in a taxonomic hierarchy
  - To define each class’ properties and include any restriction on their values
  - To assign values for each property to create instances.

- Components in ontologies (for agents)
  - Classes (descriptions of the concepts in a domain)
  - Properties (attributes and relations in classes)
  - Restrictions (data type, cardinality…)
  - Instances (constitute the concrete items/individuals represented by the ontology)
There is no single standard methodology to develop ontologies.

There is no single correct method to model a domain. Best solution depends on given application/domain.

In most methodologies the following 5 phases are present:
- Phase 1: Determine the domain and coverage for the ontology.
- Phase 2: Consider to re-use existing ontologies.
- Phase 3: Enumerate the important terms in the ontology.
- Phase 4: Define the classes and their hierarchy.
- Phase 5: Define the attributes for each class.

For more details, check the “Ontology 101” document.

Need to express ontologies in a machine-computable language (usable by agents in their messages and in their reasoning):
- A language simple enough to make ontology development easier.
- A language with formal semantics:
  - Formal semantics are needed in order to obtain deductions from the information in the ontology.
- A language allowing agents to reason with it.
- The computational cost should be reasonable.
Description Logics

- FOL + new operators and symbols
  - ⊨ (if and only if), ⊨ (if)
  - ∪ union, ∩ intersection
  - ⊤ (universal set, theorem), ⊥ (empty set, contradiction)
- Distinction between two kinds of predicates
  - Concepts (C)
  - Relations (R)
- Quantified formulae are rewritten:
  \[ \exists R.C \equiv \exists y(R(x, y) \land C(y)) \]
  \[ \forall R.C \equiv \forall y(R(x, y) \rightarrow C(y)) \]

Example

- A student, by def., is a person which has a name, an address and has registered for a course.

  \[ \text{Student} \models \text{Person} \land \exists \text{Name.String} \land \exists \text{Address.String} \land \exists \text{Registered.Course} \]

  \[ \equiv \forall x(\text{Student}(x) \rightarrow \text{Person}(x) \land \exists y(\text{Name}(x, y) \land \text{String}(y)) \land \exists z(\text{Address}(x, z) \land \text{String}(z) \land \exists w(\text{Registered}(x, w) \land \text{Course}(w))) \]

- A person should be a man or a woman.

  \[ \text{Person} \models \text{Man} \lor \text{Woman} \equiv \forall x(\text{Person}(x) \rightarrow \text{Man}(x) \lor \text{Woman}(x)) \]
Ontologies

Languages

Generic Languages

- CyCL – CPI
- KIF – CPI (Description Logic)
- CLIPS – COOL (Object Oriented)

Markup languages

- HTML
- SHOE
- XML
- SHOE
- ebXML
- XOL
- RDF-RDFS
- OIL
- DAML
- XML-S
- DAML+OIL
- OWL

Knowledge Interchange Format

- Developed at Stanford University (1992)
- Idea: to have an exchange format between applications, independent from their internal representations.
- Based in First Order Logic (FOL)
  - Prefix notation + definitions
- Semantics: Description Logics (Definitions + needed conditions)
KIF

- KIF has FOL’s operators
  - Boolean values: true, false
  - Connectives:
    - and, or, not,
    - => (if) <= (only if), <=> (definition)
  - Quantifiers: forall, exists
  - Vars: ?x (individual var) @x (var group, as in PROLOG)
- E.g., (forall (?x) (> ?x 3))
- Lists can be built and used as basic data types (as LISP)
  (listof 3 ‘HOLA’ 2.34)

Functions can be defined
(deffunction abs (?x) := ((if (> ?x 0) ?x (- ?x)))))

Relations can be defined
(deffunction number (?x) :=
  (or (integer ?x) (real ?x) (complex ?x)))
<=
  (> ?x)
  (or (integer ?x) (real ?x) (complex ?x)))

Metaknowledge expressions
(believes john ‘(exists (?x) (> ?x 3)))
**KIF example**

- **Class person**

  
  (defrelation name (?x) := (string ?x))

  (defrelation age (?x) := (integer ?x))

  (defrelation person (?x ?y) :=
   (listof (name ?x) (age ?y))
    (defobject juan:= (person “Juan” 25))

  (defrelation adult (?x) :=
   (and (= ?x (person ?x ?y))
    (> ?y 18)))

**Markup Languages: XML**

- **Idea of a Semantic Web:**
  - Information semantically annotated in a machine-parseable language

- **HTML is not enough**
  - Language oriented to presentation

- **Idea:** to use XML (derived from SGML)

- **Advantages**
  - allows to describe attributes in information
  - already used by industrial initiatives
  - allows integration from different data sources (by means of XSLT translation rules)
  - Non-proprietary language
XML

- An XML document can contain Data Type Definitions inside or can refer to a DTD file
- One can create repositories of reusable definitions (namespaces)
- E.g.:

```
<!Element direction (name, place)>
<!Element place (street, city)>
<!Element name (#PCDATA)>
...
<direction>
  <name> John </name>
  <place>
    <street> Oxford St. </street>
    <city> London </city>
  </place>
</direction>
```

From XML to DAML+OIL

- Problems:
  - XML is too rigid (tree-like structures)
  - Difficult to include relationships to the structures defined
  - Difficult to assert predicates
- Extension: RDF + RDFS
  - RDF allows to assert statements
  - RDFS declares classes, attributes and relations
  - RDFS definitions can be instantiated
- Even more powerful extension: DAML+OIL
  - DARPA agent markup language
  - Ontology Inference Layer
DAML+OIL example

```xml
<daml:Ontology>
  <daml:Class rdf:ID="Person"/>
  <daml:ObjectProperty rdf:ID="Name">
    <daml:domain rdf:Resource="http://..." />
  </daml:ObjectProperty>
  <daml:ObjectProperty rdf:ID="Parent_of">
    <daml:domain rdf:Resource="#Person" />
    <daml:range rdf:Resource="#Person" />
    <daml:Restriction daml:Cardinality="2" />
  </daml:ObjectProperty>
  <daml:ObjectProperty rdf:ID="Son_of">
    <daml:InverseOf rdf:Resource="#Parent_of" />
  </daml:ObjectProperty>
</daml:Class>
</daml:Ontology>
```

2. (Part 1) Knowledge Representation

Further extension: OWL (Ontology Web Language)
- Fusion of DAML+OIL, standard of W3C
- 3 levels:
  - OWL lite: defines taxonomies and simple restrictions
  - OWL DL: provides expressiveness as Description Logic
  - OWL full: maximum expressiveness (but not available reasoners)
References


These slides are based mainly in W3C documents, [2], [5] and material from U. Cortés and J. Bejar.