11.1 Knowledge Representation

- For expert systems and deductive databases we have used of logics in the form of basic facts and simple rules
  - Datalog/Prolog rules
  - Fuzzy reasoning
  - ...

- Is that enough to represent all real world knowledge?

11.1 Knowledge Representation

- There are several representation frameworks that differ in their degree of expressiveness
  - Simple controlled vocabularies (catalogs, glossaries)
  - Simple relations between entities (classifications, thesauri)
  - Semantic networks (ontologies, frames)
  - Logic systems (first order predicate logic, description logic)
  - Multilayered extended semantic networks (MultiNet)

11.1 Knowledge Representation

- Knowledge representation is concerned with how to formally think (or at least reason...)
  - It needs...
    - A symbol system that represents a domain of discourse
    - A formalized reasoning system to allow inference (symbol manipulation)
  - The representation instance is called a knowledge base

- There is a vast variety of suggested symbol sets, languages and inference methods...
  - With different expressiveness and different complexity
  - Rule of thumb: the more expressive, the more complex

11.1 Knowledge Representation

- Basically every representation abstracts from the real world to the domain of discourse
  - Results of reasoning processes are then reapplied to the real world
Humans reason in natural language, but this is usually very ambiguous—“The chair was placed on the table. It was broken.”
- What was broken?
- It is not represented in the sentence...
- “The dog was placed on the table. It barked.”
- Here it is clear, but still not represented in the sentence
- And anyway, which dog was it?
- Good systems need three kinds of uniqueness: referential, semantic, and functional

A necessary property of representation systems is referential uniqueness—Symbolic representations have to explicitly define relations for entity references
- All ambiguity with respect to entities must be eliminated in the internal representation
- That means that all individual entities get a unique name
- Dog-1, Dog-2, Dog-3,...
- Unique names are called instances or tokens

The second property of representation systems is semantic uniqueness—All symbols of internal representation must be unique
- This means that also word-sense ambiguity has to be resolved
- Problem of homonyms
- Money on a bank?
- To catch a ball..., to catch a thief..., to catch a cold...?

The third property is functional uniqueness—Internal representations must uniquely express the functional roles
- This includes different sentence structures and the problem of synonyms, too
- Tom catches the ball. The ball is caught by Tom...
  - Tom attends the lesson. Tom participates in the lesson, Tom took part in the lesson.
### 11.1 Mapping the Real World

- **Steps from a simple linguistic sentence to a computer-understandable representation**

```
Tom catches a ball.
Tom12 catches a ball546.
```

### 11.1 Linguistic Sentences

- **Usually a single predicate expression** is not enough to reflect the semantics of a sentence:
  - “Tom catches a white football.”
    - `catch_object(Tom, football).`
    - `instance(football, ball).`
    - `color(football, white).`
  - But with only these predicates we lose some structure:
    - The functional role within predicates is not clear.
    - The information is derived from a single sentence and may only be valid for here.
    - A football is always a ball, but not always white.

### 11.1 Linguistic Sentences

- **The slot assertion notation** assigns roles to the different arguments of predicates:
  - `catch_object(Tom, football).`
  - `catch_object(Peter, beercan).`
  - `instance(X0005, catch_object).`
  - `catcher(X0005, Tom).`
  - `caught(X0005, football).`
  - `believes(Peter, X0005).`

### 11.1 Linguistic Sentences

- **The slot and filler notation** (frames) now combines different slot assertions to provide a structured expression:
  - The resulting expression is object- or event-centered
    - `instance(X0005, catch_object).`
    - `catcher(X0005, Tom).`
    - `caught(X0005, football).`
    - `Is joined to
      - `catch_object(X0005, (catcher Tom), (caugh football)).`
11.2 The Cyc System

- **Cyc** tries to use knowledge representation as a base of building a real AI system
  - Started in 1984 by Douglas Lenat in Austin, TX
  - Best funded AI project of all times: $60 Million
  - First developed at the Microelectronics and Computer Technology Corporation (MCC), founding of Cycorp

- The aim is to create a program with enough understanding such that it can learn from books
  - To do this Cyc needs the common knowledge of a student in first grade
    - However, the world knowledge of children at this age is already quite large
    - This is especially true for understanding stories: e.g., if somebody stands ‘in front of the window’, he/she is still inside the house
    - Seems trivial, but is a lot of work to encode…

- Facts and rules are encoded by humans and tested for consistency by the system
  - If facts contradict each other, humans need to resolve them
  - After 10 years about one million rules had been encoded and half of them were used by the system
  - Many conflicts could not easily be resolved, although a small child learns to manage the inconsistencies
    - Think about fairy tales: a witch is an evil, old woman, but then witches do not exist…

- It is conjectured by the Cyc team that about 10 million rules (called assertions in Cyc) will suffice model “intelligence”
- Currently, Cyc contains around 200,000 terms, and each has several dozens of assertions
- Thus, Cyc is what is sometimes called a GOFAI, or ‘Good Old Fashioned Artificial Intelligence’
  - i.e. takes a huge set of common sense propositions and generates further propositions via inductive and deductive inference

- Cyc introduced **CycL** as an representation language for knowledge
  - Based first-order logic, syntax similar to LISP
- Using CycL, the knowledge base is created
  - Knowledge base contains classification of things (starting with the most general category: Thing)
  - Divided in thousands of ‘microtheories’
    - Microtheories contain a bundle of assertions sharing a set of common assumptions
    - Focus on either a certain domain, a particular level of detail, or time interval, etc

- General knowledge: things, intangible things, physical objects, individuals, collections, sets, relations…
  - Domain-specific knowledge, for example:
  - Political geography; general information (e.g. What is a border?) and specific information about towns, cities, countries and international organizations Human anatomy and physiology
  - Chemistry
  - A lot of others - see Cycorp web page http://www.cyc.com/
11.2 The Cyc System

• What does Cyc contain?

- Knowledge-Based Systems and Deductive Databases
- Wolf - Tilo Balke
- IfIS - TU Braunschweig

11.2 The Cyc System

• The Cyc system comes in two flavors
  - The “real” Cyc
    - Contains all terms, assertions, and also reasoning capabilities
  - OpenCyc
    - Open Source
    - Contains all terms, and all basic assertions
      - No proprietary assertions
      - No reasoning
    - Open Cyc available as CycL knowledgebase or as “semantic web edition” either online or as OWL
      - Web edition provides permanent endpoints for terms

11.2 The Cyc System

• OpenCyc knowledge base web entry for „student“
  - http://sw.opencyc.org/concept/Mx4rVrIgswEBdiC0S9ycA

11.2 The Cyc System

• OpenCyc knowledge base web entry for „Kurt Gödel“
  - http://sw.opencyc.org/concept/Mx4rvarfZSwEbdvC0S9ycA

11.2 The Cyc System

• However, due to its inductive nature Cyc has several problems
  - Conclusions are only valid as long as all assertions and facts are valid
    - Hidden inconsistencies break the reasoning process
  - “Animals are dumb”, “Humans are animals”, “Scientists are humans who develop scientific theories”, “To develop scientific theories, one needs to be clever”
    - ⇐ “There are no scientists”

11.2 The Cyc System

• Also, even consistent data will lead to false conclusions when the dataset is unrepresentative for the real world
  - At one point, Cyc contained many famous persons
    - Like the dead organism Gödel…
  - Based on this knowledge, it reasoned that all people are famous
    - … because each person it knew was famous
    - So, from Cyc’s viewpoint this deduction was completely valid
What can Cyc be used for?

- **Directed marketing**
  - Use a person’s buying history to infer their hobbies, interests, occupation, physical needs and preferences, etc. From that model, decide which products to try to sell them, and what “argument” to use to convince them they should buy the product.

- **Data base integration**
  - Use that same sort of “articulation” approach to have several heterogeneous data bases all relate their contents to one central knowledge base.

- **Machine translation of technical documents**
  - Use Cyc to interpret the words encountered in natural language to generate a meaningful translation

- **Language understanding**
  - As above, use to understand certain texts and mine for its meaning

- **Security**
  - One of Cyc’s major sponsors…
  - CycSecure is able to scan security relevant documents and network traffic in order to identify potential security risks or breaches

**11.2 The Cyc System**

**11.3 Knowledge on the Web**

- Representing knowledge is also useful for annotating information on the Web
  - Web pages are uniquely identified by their **URLs**
  - Information like the author, change date, etc. of a Web site can be represented
  - Remember: the WWW started as a telephone directory at CERN

- The current Web **represents** information using
  - Natural language
  - Graphics, multimedia, page layout

- Humans can process this **easily**…
  - Can deduce facts from partial information
  - Can create mental associations
  - Are used to various sensory information

- Information is **searched** by using techniques from information retrieval (IR) or page ranking
  - Still, the information on a page is not understood

- **Tasks** often require to combine data on the Web
  - Hotel and travel information may come from different sites
  - Searches in different digital libraries

- Again, humans **combine** these information quite easily
  - even if different terminologies are used

- What is needed?
  - Data should be available for **machines** for further processing
  - Data should be possibly **combined**, or **merged** on a Web scale
  - Often, data may describe **other** data

- Moreover, the data is to be exchanged by itself, like my calendar or my travel preferences
  - Machines may also need to **reason** about that data
11.3 Knowledge on the Web

• Still, although Web sites are uniquely identifiable, their information is not.
  – But if items on or in relation to Web sites are identified and assigned a unique identifier, we can properly represent knowledge.
  – So-called URIs (uniform resource identifier).

11.3 The Semantic Web

• The goal of the semantic web is not to understand natural language, but provide web pages in an computer readable form.
  The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing.

11.3 Building Blocks

• The architecture of Semantic Web technology:
  – So-called Semantic Web stack.
  – We will work our way through from the bottom up.
  The architecture of the Semantic Web stack.

11.3 Encoding Knowledge

• All resources are described by the resource description framework (RDF).
  – Became a W3C recommendation in 2004.
  – Idea: use XML to define metadata, i.e. data about data.

• The RDF standard deals with two topics:
  – The abstract model: how to encode knowledge?
    • http://www.w3.org/TR/rdf-concepts/
  – How to denote such statements in XML?
    • http://www.w3.org/TR/rdf-syntax-grammar/
11.3 Encoding Knowledge

• The graph structure...

\[
\text{Bruce Willis} \quad \text{played_in} \quad \text{Die Hard} \\
\text{Die Hard} \quad \text{is_a} \quad \text{Movie} \\
\text{Die Hard} \quad \text{released_in} \quad 1988
\]

• …Can be broken down to triples

<table>
<thead>
<tr>
<th>Start node</th>
<th>Edge label</th>
<th>End node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bruce Willis</td>
<td>played_in</td>
<td>Die Hard</td>
</tr>
<tr>
<td>Die Hard</td>
<td>is_a</td>
<td>Movie</td>
</tr>
<tr>
<td>Die Hard</td>
<td>released_in</td>
<td>1988</td>
</tr>
</tbody>
</table>

11.3 Encoding Knowledge

• These triples are often referred to as subject, predicate, and object
  – A fact is expressed as a Subject-Predicate-Object triple, also known as a statement
  – Names of subjects and predicates are URIs
    • Global in scope: always referring to the same entity in any RDF document in which they appear
  – Names of objects can be URIs, but also be given as text values, called literal values, which may or may not be typed
    • E.g., Die Hard’s release date 1988 is a literal value

<table>
<thead>
<tr>
<th>Start node</th>
<th>Edge label</th>
<th>End node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject</td>
<td>predicate</td>
<td>object</td>
</tr>
</tbody>
</table>

11.3 Encoding Knowledge

• A quick note on URIs in RDF
  – RDF uses URI references which are URIs, together with an optional fragment identifier at the end
    • The URI reference http://www.w3c.org/index#section2 consists of the URI http://www.w3c.org/index and the fragment identifier section2 (separated by the # character)
  – Since URIs can be quite long, in RDF notations they are usually abbreviated using the concept of namespaces from XML
    • A namespace with a certain URI can be assigned to a prefix
      • E.g., prefix: w3c is assigned URI: http://www.w3c.org/
      • Now w3c:index means http://www.w3c.org/index

11.3 Encoding Knowledge

• But consider that some information has a structure provided by literals about a resource
  – This leads to structured property values, like e.g., addresses consisting of streets, cities, Zip codes
  – One possibility is to create an intermediate node representing the concept of the structure
    • Such a node would need a new URI
    • All literals concerning the concept would connect to this URI, but otherwise it is rather useless…
  – The second possibility is to use a blank node or anonymous resource

11.3 Encoding Knowledge

• But for storing blank nodes in triples, they need a unique name, so-called blank node identifiers

<table>
<thead>
<tr>
<th><a href="http://www.example.org/bruce_willis">http://www.example.org/bruce_willis</a></th>
<th>blank node identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.org/die_hard">http://www.example.org/die_hard</a></td>
<td>blank node identifier</td>
</tr>
</tbody>
</table>

11.3 RDF/XML

• Remember: resources are identified by URIs and described in terms of properties and values
  – http://www.example.org/bruce_willis identifies a person whose name is ‘Bruce Willis’ and who starred in the movie ‘Die Hard’
**11.3 RDF/XML**

- **To represent statements in a machine-processable way RDF uses XML**
  - It defines a specific XML markup language RDF/XML

  ```xml
  <?xml version="1.0"?>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://www.example.org/" ns#="http://www.example.org/" xml:lang="en">
    <ex:movie rdf:resource="http://www.example.org/die_hard"/>
    <ex:starred_in rdf:about="http://www.example.org/bruce_willis">
      <ex:full_name>"Bruce Willis"</ex:full_name>
    </ex:starred_in>
    <ex:born_in rdf:resource="http://www.example.org/bruce_willis">
      <ex:full_name>"Bruce Willis"</ex:full_name>
    </ex:born_in>
  </rdf:RDF>
  ```

- **RDF applications sometimes need to describe other RDF statements using RDF**
  - E.g., to record when statements were made, who made them, ... so-called provenance information
  - RDF provides a built-in vocabulary intended for describing RDF statements
    - A description of a statement using this vocabulary is called reification of the statement

- **The RDF reification vocabulary consists of**
  - the type rdf:Statement
  - the properties rdf:subject, rdf:predicate, and rdf:object
  - For example we might be interested in who actually provided Bruce Willis' year of birth

  ```xml
  <rdf:Statement rdf:about="http://www.example.org/bruce_willis">
    <ex:born_in rdf:resource="#triple15667">"1955"</ex:born_in>
  </rdf:Statement>
  ```

- **Using reification needs careful considerations**
  - The subject of the reification triples should identify a particular instance of a triple in a particular RDF document
    - There could be several triples that have the same subject, predicate, and object, but RDF provides no way to distinguish between them
    - This can be (to some degree) remedied by using triple identifiers directly in the predicate of some instance

  ```xml
  <ex:movie rdf:resource="#triple15667">"Bruce Willis"</ex:movie>
  <ex:born_in rdf:resource="#triple15667">"1955"</ex:born_in>
  ```
### 11.3 RDF/XRL

- For more details on the exact syntax see W3C’s **RDF/XML Syntax Specification**
  - Accepted as W3C recommendation in 2004
  - [http://www.w3.org/TR/rdf-syntax-grammar/](http://www.w3.org/TR/rdf-syntax-grammar/)

### 11.3 RDF Schema

- **RDF Schema** defines valid RDF statements
  - Officially called **RDF Vocabulary Description Language**
  - Provides the basic elements for the description of vocabulary ontologies
- Why do we need vocabularies?
  - RDF itself allows the use of any URI reference
    - e.g. `movie:played_in` or `mdb:acts_in`
  - However, often it is beneficial to restrict the possible terms and/or relations in order to increase interoperability between RDF documents
    - Think of integration...

### 11.3 RDF Schema

- Without controlled vocabulary, shared domain ontologies are hard to realize
  - `starred_in` \(\equiv\) `acts_in` ??

### 11.3 RDF Schema

- First of all, RDF vocabulary written in RDF-S are valid RDF graphs themselves
  - RDF-S mainly provides predefined edge labels and end labels
- **RDF Classes**
  - All used resources can be divided into groups called “classes”
    - Similar to classes in OO languages like Java
    - Classes may have subclasses (and thus inheritance)
    - Membership to a class is indicated by `rdf:type`, subclasses by `rdfs:subClassOf`

### 11.3 RDF Schema

- Example:
  - `mdb:Movie` `rdf:type` `rdfs:Class`
  - `mdb:Comedy` `rdfs:subClassOf` `mdb:Movie`
  - `mdb:ActionMovie` `rdfs:subClassOf` `mdb:Movie`
    - i.e. Movie is a class of specific “things”, ActionMovies are specialized movies (thus each ActionMovie is also a Movie)
- The characteristics of classes can be described using **properties**
  - Technically, properties are also classes but are only used to further describe their associated base classes
11.3 RDF Schema

• RDF-S provides further facilities to describe the nature of properties and classes
  – Like its range, domain, type, etc.

• Example:
  – mdb:Movie rdf:type rdfs:Class
    mdb:Person rdf:type rdfs:Class
    mdb:has_cast rdf:type rdfs:range mdb:Person
    mdb:has_cast rdf:domain rdfs:Class

11.3 RDF Schema

• Using RDF-S within a RDF knowledgebase:
  – Instances are usually defined within their own document
  – Schema is imported

  `<html version="1.0">`
  `<xml RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"`
  xmlns: mdb="http://www.mdb.org/schemas/movieSchema">`
  `<mdb:Movie rdf:id="DieHard">`
  `<mdb:hasCast rdf:resource="mdb:BruceWillis" />`
  `<mdb:Movie>`
  `<mdb:Person rdf:id="BruceWillis" />`
  `</rdf:RDF>`

11.3 RDF Schema - Classes

<table>
<thead>
<tr>
<th>Element</th>
<th>Class of</th>
<th>rdfs:subClassOf</th>
<th>rdf:type</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:Resource</td>
<td>rdf:Resource</td>
<td>rdfs:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Class</td>
<td>rdf:Resource</td>
<td>rdfs:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Literal</td>
<td>rdf:Literal</td>
<td>rdfs:Literal</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Statement</td>
<td>rdf:Statement</td>
<td>rdfs:Statement</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:List</td>
<td>rdf:Resource</td>
<td>rdfs:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Bag</td>
<td>rdf:Resource</td>
<td>rdfs:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Seq</td>
<td>rdf:Container</td>
<td>rdfs:Container</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Alt</td>
<td>rdf:Container</td>
<td>rdfs:Container</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Container</td>
<td>rdf:Alt</td>
<td>rdfs:Alt</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:Membership</td>
<td>rdf:_</td>
<td>rdfs:property</td>
<td>rdf:Property</td>
</tr>
</tbody>
</table>

11.3 RDF Schema - Properties

<table>
<thead>
<tr>
<th>Element</th>
<th>Relates</th>
<th>rdfs:domain</th>
<th>rdfs:range</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:range</td>
<td>restricts</td>
<td>rdf:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:domain</td>
<td>restricts</td>
<td>rdf:Property</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:type</td>
<td>instance of</td>
<td>rdf:Resource</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:subClassOf</td>
<td>subclass of</td>
<td>rdf:Class</td>
<td>rdf:Class</td>
</tr>
<tr>
<td>rdf:subPropertyOf</td>
<td>property of</td>
<td>rdf:Property</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdf:label</td>
<td>human readable</td>
<td>rdf:Resource</td>
<td>rdf:Literal</td>
</tr>
<tr>
<td>rdf:comment</td>
<td>human readable</td>
<td>rdf:Resource</td>
<td>rdf:Literal</td>
</tr>
<tr>
<td>rdf:member</td>
<td>container membership rdf:Resource rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:seeAlso</td>
<td>further information rdf:Resource rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:isDefinedBy</td>
<td>definition rdf:Resource rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:hasValue</td>
<td>for structured values rdf:Resource rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:subject</td>
<td>subject of statement rdf:Statement rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:predicate</td>
<td>predicate of statement rdf:Statement rdf:Resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rdf:subject</td>
<td>subject of statement rdf:Statement rdf:Resource</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.3 RDF Schema

• Note: Properties in RDF-S differ slightly from those used in OO languages
  – OO:
    • Properties are part of the class definition
  – RDF-S:
    • Properties are defined on their own and assigned to classes via their range and domain.
    • Multiple ranges and/or domains are possible
      – e.g. has_cast might have two ranges, namely mdb:Movie and tdb:TheaterPerformance
    • Data types, typed literals, comments, labels, enumerations, references, etc.
      – See next slides…

    – The full standard is available under:
      http://www.w3.org/TR/rdf-schema/

11.3 RDF Schema - S

• RDF-S provides much additional semantics for specifying schemas
  – data types, typed literals, comments, labels, enumerations, references, etc.
    – See next slides…

Knowledge Systems and Database Systems - 11.3 RDF Schema - 61 - Tilo Balke

11.3 RDF Schema - S

• RDF-S provides much additional semantics for specifying schemas
  – data types, typed literals, comments, labels, enumerations, references, etc.
    – See next slides…

    – The full standard is available under:
      http://www.w3.org/TR/rdf-schema/

Knowledge Systems and Database Systems - 11.3 RDF Schema - 65 - Tilo Balke

11.3 RDF Schema - S

• RDF-S provides much additional semantics for specifying schemas
  – data types, typed literals, comments, labels, enumerations, references, etc.
    – See next slides…

    – The full standard is available under:
      http://www.w3.org/TR/rdf-schema/

Knowledge Systems and Database Systems - 11.3 RDF Schema - 71 - Tilo Balke
• Taxonomies & Ontologies
  – Knowledge representation
  – Reasoning
  – DAML+OIL
  – OWL