Semantic Web & Ontology – SWE 2022 [~KShawki]

***Semantic Web***

Providing common machine-readable framework that allows data to be shared and reused across application, enterprise, and community boundaries.

* Allowing machines to understand data
* Ease sharing and mixing data
* Extend the World Wide Web rather than replace it

**Semantic Web:** An extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

**What is Semantic Web for?**

* *Integrating:* trying to solve the problem of data and service integration
* *Searching:* Providing better communication between human and computers by adding machine processable semantics to data.

**Ontology:** describes the individual instances and roles in the domain that are represented using unary and binary predicates.

**Evolution of World Wide Web**

1. Desktop Computing [1980-1990]
2. Web 1.0 [1990-2000].
3. Web 2.0 [2000-2010].
4. Web 3.0 [2010-2020].

**World Wide Web Basic Ideas:**

* Hypertext/hyperlink:
* Resource Identifiers
  + Unique identifiers used to locate a particular resource (computer file, document or other resource) on the network
  + URI (Uniform Resource Identifier) / URL (Uniform Resource Locator): http or ftp
* Markup Language:
  + Characters or codes embedded in text which indicate structure, semantic meaning, or advice on presentation

**Web 2.0**

“Web 2.0 has come to refer to what some people describe as a second phase of architecture and application development for the World Wide Web.” ~Wikipedia.

* The Web where "ordinary" users can meet, collaborate, and share using social software applications on the Web (tagged content, social bookmarking, AJAX, etc.)

Features:

1. *The Web as platform*
2. *Harnessing collective intelligence*
3. *Data is the next "Intel Inside"*
4. *End of the software release cycle*
5. *Lightweight programming models*
6. *Rich user experiences*
7. *Software above the level of a single device*

***Knowledge Engineering***

An engineering discipline that involves integrating knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise.

* It normally involves five distinct steps in transferring human knowledge into some form of knowledge-based systems (KBS)
  + Identify the task
  + Assemble the relevant knowledge
  + Decide on a vocabulary of predicates, functions and constants
  + Encode general knowledge about the domain
  + Encode a description of the specific problem instance
  + Pose queries to the inference procedure and get answers
  + Debug the knowledge base

**Knowledge Engineering Process**

* Acquisition of knowledge
  + General knowledge or metaknowledge
  + From experts, books, documents, sensors, files
* Knowledge representation
  + Organized knowledge
* Knowledge validation and verification
* Inferences
  + Software designed to pass statistical sample data to generalizations
* Explanation and justification capabilities

**Terminology**

* **Domain:** some area of interest
* **Task:** something that needs to be done by an agent.
* **Agent:** the executor of a task in a domain.
* **Application:** the context provided by the combination of a task and a domain in which this task is carried out by agents.
* **Application domain:** The particular area of interest involved in an application.
* **Knowledge:** is the sort of information that people use to solve problems.
* **Knowledge System [KS]:** system that solves a real-life problem using knowledge about the application domain and the application task.
* **Knowledge Based System [KBS]:** Software system, which represents (explicit, declarative description of knowledge) and uses this knowledge to accomplish a task within the context of a certain application
  + [KBS = knowledge-base + inference engine]

**KBS Stockholders**

* **Domain expert:** The individual or group whose expertise and knowledge is captured for use in an expert system
* **Knowledge user:** The individual or group who uses and benefits from the expert system
* **Knowledge engineer:** Someone trained or experienced in the design, development, implementation, and maintenance of an expert system.

**What is Knowledge?**

* *The facts, feelings, or experiences known by a person or group of people.*
* **Knowledge includes** facts, concepts, procedures, models, heuristics, examples.
* **Knowledge may be** specific or general, exact or fuzzy, procedural or declarative

**Data, Information, and Knowledge**

* **Data:** Unorganized and unprocessed facts; static; a set of discrete facts about events.
* **Information:** Aggregation of data that makes decision making easier
* **Knowledge** is derived from information in the same way information is derived from data; it is a person’s range of information.

**Main types of KBS:**

|  |  |  |
| --- | --- | --- |
| * Expert systems * Neural networks. * Case-based reasoning. | * Genetic algorithms * Intelligent agents | * Data mining * Intelligent Tutoring systems. |

**Taxonomies of Knowledge**

* *Five Types of Knowledge*
  + Declarative knowledge - **Know-about**
  + Procedural knowledge - **Know-how**
  + Causal knowledge - **Know-why**
  + Conditional knowledge - **Know-when**
  + Relational knowledge - **Know-with**
* *Meta-knowledge*
  + Knowledge about knowledge

**Problems in knowledge engineering**

* Complex information and knowledge are difficult to observe.
* Experts and other sources differ.
* Multiple representations.

**Other Knowledge Representation**

* Semantic Net
* Frames
* Rule based systems

***Semantic Net:***

Graphical way for representing declarative knowledge.

* It is a labeled, directed graph consisting of vertices and edges.
* Knowledge represented as a set of nodes connected by labeled arcs
* Nodes stand for objects, concepts, attributes or events
* Arcs represent relationships between the nodes
* Good for:
  + Semantic associations (taxonomies)
  + Physical and causal structures
* Object representation
  + Two kinds of node to represent objects: classes and instances (objects).
  + Two kinds of relationships between class/objects:
* IsA relates an instance to its class
* AKindOf relates a sub-class to its superclass

**Types of Relations:**

* **IS-A:** relates an instance (individual node) to a class (generic node)
* **AKO (a-kind-of):** relates one class (subclass) to another class (superclass).

**Advantages of Semantic Nets:**

* Easy to visualize
* Formal definitions of semantic networks have been developed.
* Related knowledge is easily clustered.
* Efficient in space requirements
  + Objects represented only once.

**Disadvantage of Semantic Nets:**

* Inheritance (particularly from multiple sources and when exceptions in inheritance are wanted) can cause problems.
* No standards about node and arc values.

**Inference in a Semantic Net**

Basic inference mechanism: follow links between nodes.

Two methods to do this:

* **Intersection search**
  + find relationships between objects by spreading activation from each of two nodes and seeing where the activations met. This process is called intersection search.
* **Inheritance**
  + The AKO, isa and instance representation provide a mechanism to implement this.

***Object-Oriented representation***

Objects: a natural way to organize the knowledge about:

* Physical objects: a desk has a surface-material, # of drawers, width, length, height, color, procedure for unlocking, etc.
* Situations: A classroom: participants, teacher, day, time, seating arrangement, lighting, procedures for registering, grading, etc.
* A trip: destination, origin, conveyance, procedures for buying ticket, getting through customs, reserving hotel room, etc.
* Important: Objects enable grouping of procedures for determining:

– properties of objects, their parts, interaction with parts

***Frames***

A frame is viewed as a data structure for representing stereotyped situations to which are attached various kind of information including the object or events.

* A frame system attempts to integrate
  + Declarative notions about objects and events and their properties.
  + Procedural notions about how to retrieve information and achieve goals.

**Knowledge Based System**

The goal is to facilitate intelligent interaction with the user based on:

* the identification of the appropriate information
* the effective utilization of the appropriate information
* the control of the appropriate information

in order to fulfill specific user goals

**Expert Systems**

An expert system is a computer system that emulates, or acts in all respects, with the decision-making capabilities of a human expert.

**Rule-Based System**

* Rules can be used to formulate a theory of human information processing
* rules are stored in long-term memory (KB)
* temporary knowledge is kept in short-term memory (agenda)
* Input or thinking triggers the activation of rules
* activated rules may trigger further activation
* a cognitive processor combines evidence from currently active rules
* This model is the basis for the design of many rule-based systems
* also called production systems.

**Chaining**

a group of multiple inferences that connect a problem with its solution.

* Two kinds of control in rule-based systems:
  + **Forward chaining:** A chain that is searched / traversed from a problem to its solution.
  + **Backward chaining:** A chain traversed from a hypothesis back to the facts.
  + Problem with backward chaining is find a chain linking the evidence to the hypothesis

**Forward chaining (data-driven)**

* Forward chaining is bottom-up reasoning, i.e. reasoning from facts to goals. Reasoning (moving) from facts to the conclusion
* As soon as facts are available, they are used to match antecedents of rules
* This process of cascading triggering of rules is called ‘chaining’ as a chain of rules may be fired.

**Backward chaining (goal/query-driven)**

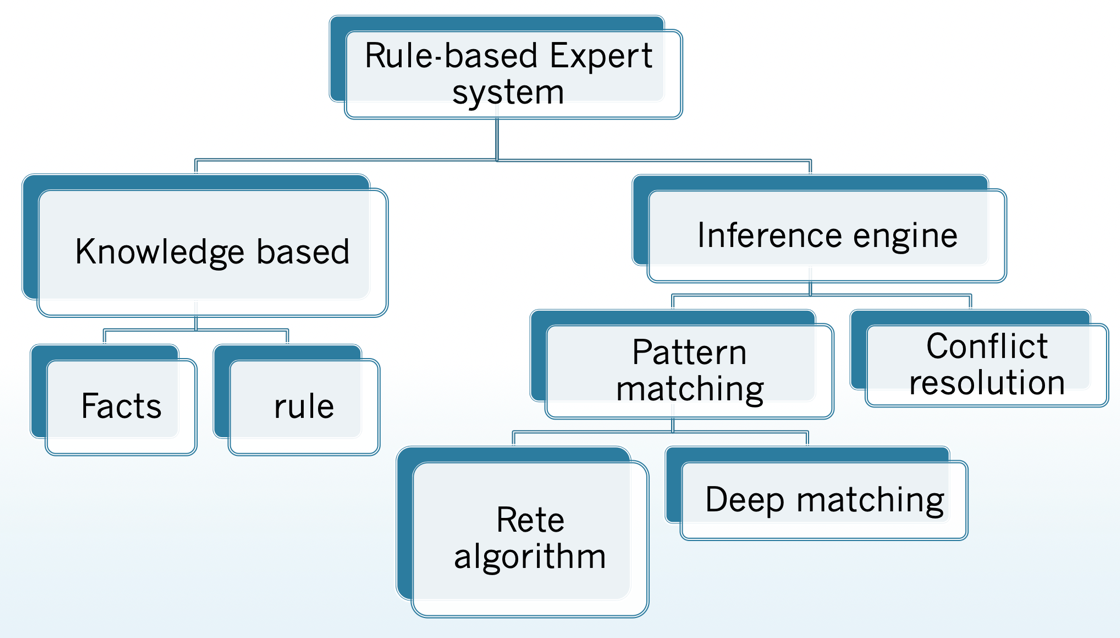
* Backward chaining is top-down reasoning, (reasoning from goals to facts).
* Starting from a hypothesis (query-goal, supporting rules and facts are sought until all parts of the antecedent of the hypothesis are satisfied)
* often used in diagnostic and consultation systems

|  |  |  |
| --- | --- | --- |
| Comparison | **Forward chaining** | **Backward chaining** |
| planning, control | diagnosis |
| data-driven | goal-driven (hypothesis) |
| Bottom-up reasoning | Top-down reasoning |
| *find possible conclusions supported by given facts* | *find facts that support a given hypothesis* |
| like breadth-first search | like depth-first search |
| antecedents (LHS) control evaluation | consequents (RHS) control evaluation |

**Advantages of Rule Systems**

* Separation of control (inference) from knowledge.
* Modularity of knowledge (A rule is an independent chunk of knowledge).
* Ease of expansion (Proportional growth of intelligence)
* Derivation of explanations from rigid syntax
* Consistency checking
* Utilization of uncertain knowledge
* Can incorporate variables

**Foundations of Expert Systems**



Week 5: Intro to ontology

**Week 6: Intro to Onto.**

**XML = node = label + contents**

<module>

<title>Internet and World Wide Web</title>

<code>G52IWW</code>

<students>Undergraduate</students>

</module>

**What is an ontology?**

An ontology is a formal, explicit specification of a shared conceptualization

Chart, bubble chart

Description automatically generated

**Ontology Example**

* **Concept**: conceptual entity of the domain
* **Attribute**: property of a concept
* **Relation**: relationship between concepts or properties
* **Axiom (constraints):** coherent description between Concepts / Properties / Relations via logical expressions

**Why Develop an Ontology?**

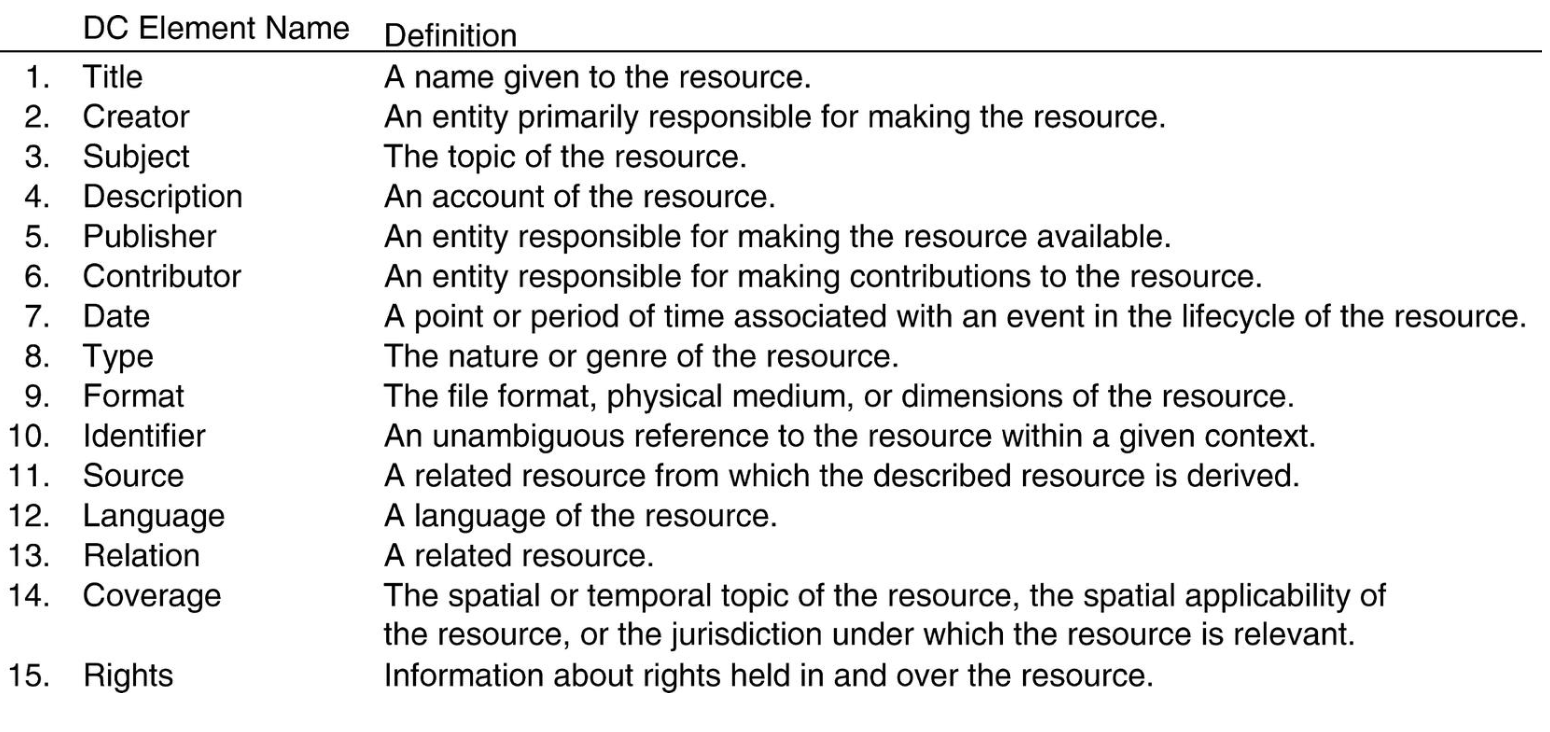
* To share common understanding of the structure of information
* To enable reuse of domain knowledge

**Ontology** **Components**:

* **Concepts**: set of entities within a domain.
* **Individuals**: instances or objects
  + concrete examples on concepts in a domain.
* **Relations**: interactions between concepts or concepts’ properties (attributes, slots).
* **Axioms**: explicit rules to constrain the use of concepts.

**Dublin Core**

A set of fifteen basic properties for describing generalized Web resources



**RDF Statement (Triple)**

* **Subject** of an RDF statement is a resource
* **Predicate** of an RDF statement is a property of a resource
* **Object** of an RDF statement is the value of a property of a resource

**Ontology Languages:**

RDF:

RDF-S: Stands of RDF-Schema

* RDFS is recognisable as an ontology language
* Classes and properties
* Sub/super-classes (and properties)
* Range and domain (of properties)

OWL: it’s combined of two languages: (OIL, DAML)

**Ontologies main stages (How to build ontology?):**

* determine domain and scope
* enumerate important terms
* define classes and class hierarchies
* define slots
* define slot restrictions (cardinality, value-type)
* Slot-cardinality: Borders\_with multiple, Start\_point single.
* Slot-value type: Borders with Country.

The RDF Graph

* An RDF document is an unordered collection of triples.
* The subject of one triple can be the object of another.
* The result is a directed, labelled graph.
* A triple’s object can also be a literal, e.g., a string.

**SPARQL**

**A SPARQL query** consists of three parts:

* **Graph pattern:** Specifying a graph pattern, which is just RDF using some variables.
  + Pattern matching: optional, union, filtering, .
* **Matching:** When RDF data matches a specific graph pattern.
  + Solution modifiers: projection, distinct, order, limit, offset, . . .
* **Binding:** When a specific value in RDF is bound to a variable in a graph pattern.
  + Output par t: construction of new triples, . . ..

**SPARQL** => Consists of two parts: query language and protocol.

**Types of SPARQL queries**

SPARQL has four query forms

* SELECT: return all, or a subset of, the variable bound in a query pattern match
* CONSTRUCT: returns an RDF graph constructed by substituting variables in a set of triple templates
* ASK: returns a boolean indicating whether a query pattern matches or not.
* DESCRIBE: Find all statements in the dataset that provide information about the following resource(s)

**PREFIX keyword:** describes prefix declarations for abbreviating URIs.

|  |
| --- |
| PREFIX foaf: <http://xmlns.com/foaf/0.1/> SELECT ?name  WHERE { ?person foaf:name ?name } |

**SELECT query** serves very much the same function in SPARQL as in SQL → to return data matching some conditions.