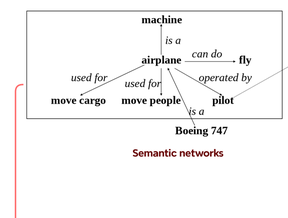
Knowledge Base Systems

For more MindMaps
Check our Biliz Website
https://biliz.devopix.com/

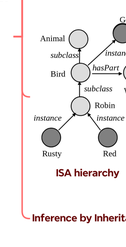
Created by
Omar Alshaheb

Introduction

- **AI Definition**
  - The design and study of computer systems that behave intelligently — deal of **knowledge** about the world
  - It depends on
    - Represent the knowledge about the world
    - Reason with the knowledge to obtain meaningful answer
- **Knowledge Representation**
  - is the study of
    - how knowledge and facts about the world can be represented
    - what kinds of reasoning can be done with that knowledge
  - the representation should be
    - rich enough to express the knowledge needed to solve the problem
    - as close to the problem as possible
      - compact
      - natural
      - maintainable
    - able to express features of the problem — we can exploit for computational gain
    - able to trade off accuracy and computation time
    - set of sentences that describe the world and its behavior — in some formal (representational) language
- **Knowledge-Based Agents**
  - consists of
    - Knowledge base
      - Typically domain specific — but large knowledge corpuse are built to provide general knowledge resources — like Cyc
    - Inference engine
      - set of procedures that use the representational language
        - to infer new facts from known ones or answer a variety of KB queries
        - Inferences typically require **search**
      - Typically domain independent
  - **Benefits of Explicit Representation**
    - We can add new tasks and easily make them depend on previous knowledge
    - Extend the existing behavior by adding new beliefs
    - Debug faulty behavior by locating the erroneous beliefs
    - Explain and Justify the behavior of the system
  - **Benefits of Reasoning** — Reasoning helps us derive
- **Production Systems**
  - **forward-chaining** — C1, C2, .. On => A1 A2 .. Am
    - Left hand side (LHS) — Conditions/antecedents
    - Right hand side (RHS) — Conclusions/consequences
    - Condition which must hold before the rule can be applied
    - Actions to be performed or conclusions to be drawn when the rule is applied
  - **basic components of production systems**
    - (Unordered set of user-defined "if-then" **rules** — Form of rules: if P1 .. Pm then A1 ,.. An
    - **Facts** determine when rule is applicable
    - **Actions** can add or delete facts from the Working Memory
    - **Working Memory**
      - Set of facts — defining what's known to be true about the world
      - inference may add/delete a fact from WM
      - WM will be cleared when a case is finished
    - **Inference Engine**
      - Procedure for inferring changes — to Working Memory
      - Can be both forward and backward chaining
      - Usually a cycle of these phases
        - **match**
        - **conflict**
        - **resolution**
        - **action**
        - in this order
      - **Conflict Resolution** Strategies
        - **Refraction** — rule can only be used once — with the same set of facts in WM
        - **Recency** — Use rules that match the facts that were added most recently to WM
        - **Specificity** — Use the most specific rule
        - **Explicit priorities** — select rules by their pre-defined order/priority
- **Structural Representations**
  - capture the interrelations between pieces of knowledge
  - Put structures into KB
  - Center around object/classes
  - More for what it is than what to do
  - The **graphical depiction** associated with a semantic network is a big reason for their popularity
  - as we can notice from the graphs on the left
    - **Nodes** — for words — denote objects/classes
    - **Directed links** — for relations/associations between words — define binary relationships between objects

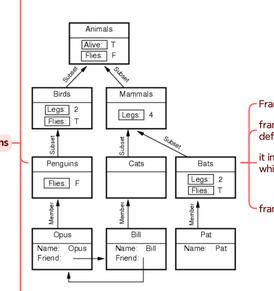


Semantic networks



ISA Nierarchy

- ISA stands for "Is a" — relation is often used to link a class and its superclass
- **Individuals & Classes** — nodes representing individuals and those representing classes
- **subclass** — is represented by "instance-of" relation
- **Inference by Inheritance**
  - inheritance of values (properties) along the subclass and instance links
  - but semantic networks differ in how they handle the case of inheriting multiple different values
    - All possible properties are inherited
    - Only the "lowest" value or values are inherited



Frame systems

- Frame is a lot like the notion of an object in OOP — but has more meta-data
- frame represents a stereotypical/expected default view of an object
- It includes the object node and all other nodes — organized in a record like structure which directly related to that object
- frame has a set of **slots**
  - each represents a relation to another frame (or value)
  - slot has one or more **facets** — each represents some aspect of the relation
- **Description Logic**
  - a family of Frame-like KR systems with a formal semantics — ex: KL-ONE
  - Subset of **FOL** (First Order Logic)
  - **Inference**
    - **Subsumption** — B subsumes A if A is an instant or a subclass of B
    - **Classification** — x is classified in A if A is the most specific subsumer
  - Keep the language **simple**
    - so that all inference can be done in polynomial time
    - ensuring tractability of inference
- Semantic Web

Production Systems

- Knowledge Engineering (KE) — developing (& maintaining) a knowledge-based system
- key aspect is Knowledge Acquisition (KA)
  - Getting the rules and heuristics — & their confidence levels
  - Best to have the participation of a live expert
  - **KA techniques**
    - **Interviewing** — Just ask the expert how they solve problems in the domain
    - **Protocol analysis** — Have the expert 'talk aloud' as they solve a problem
    - In addition to reviewing documents
  - **KA synthesis and verification**
    - **Process diagrams** — make a flowchart of the process that's followed for key decisions
    - **Ontology formulation** — collect all the concepts in the domain into a database
    - **Create influence diagrams** — show the relationship of factors to one another to actions — and to outcomes
    - Once you've identified some initial concepts and rules — begin formalisation and review with experts
- **Production Rule System**
  - emulates human reasoning using a set of 'productions
  - Productions have two parts
    - "IF" part — Sensory precondition
    - "THEN" part — Action
  - When the state of the 'world' matches the IF part — the production is fired — meaning the action is executed
  - So production rules link facts — to conclusions
  - It consists of
    - **Knowledge Base** — aka rule base
    - **Database** — contains facts
    - **Rule Interpreter** — also called a rule application module — to control the entire production system
- **Propositional logic conceptualization** — C1 & C2 & .. & Cn => A — where Cn are atomic formulas — propositional variables or them & their negations
- **Predicate Form conceptualization** — C1 & C2 & .. & Cn => A — where On are atomic formulas
  - represent records in the database
  - written in a triple form — (ID, attribute, value of the attribute)
  - or predicate form — attribute (ID, value of the attribute)
- **Production System ES** (Expert System)
  - **ES = (R, RI, DBF)**
    - R — finite set of production rules
    - RI — inference engine called **rule interpreter** — have many steps of working
    - DBF — database of facts — changes in the process
    - fire this
    - checks this
- **Pattern Matching** — **Unification** — is matching the variable x — with a **proper record** in the database
  - in the triple (x, attribute, value) — after unification we have new facts added to the DBF
  - aka attribute(x, value)
- **Conflict Resolutions** — I talked about them in the introduction
- **Meta Rules** — Rules about Rules — can be
  - domain specific
  - domain free
- **advantages of Production Systems**
  - modularity
  - Rules are independent pieces of knowledge — so may be added or deleted
- **disadvantages of Production Systems** — inefficiency of big production systems — with non-organized rules
- **forward chaining**
  - aka data-driven — aka bottom up
  - steps — During the SELECTION step of each cycle
    - the RI is looking for applicable rules
      - by MATCHING (unifying) condition part of a rule
      - with the CURRENT CONTENT of the DB
    - Forward chaining is applied — the proper rule is FIRED — and a new FACT (action part) is added to the DB
  - Process TERMINATES when the GOAL is reached — or when all possible FACTS are already inferred from the INITIAL database
- **backward chaining**
  - aka goal-driven
  - steps
    - Start with the goal
    - Applicable RULES — are found by matching ACTION parts with the GOAL, C1^ .. ^Cn => GOAL
    - Now the conditional part C1 ^ .. ^Cn is checked against the DB
    - If all are (after matching) in DB — the solution is reached
    - If Ci is not in DB — we treat it as a SUBGOAL and repeat

Object-Oriented Representation

- a little different from the lecture
- **Background & Foundational Concepts**
  - **flat representation**
    - each sentence is self contained
    - independently understood
    - info scattered in multiple sentences
  - **object oriented features**
    - sentences grouped
    - structured & organized
    - corresponded with user interface
    - data abstraction
    - inheritance
  - **frames (Minsky 1974)**
    - structure selected from memory for new situation
    - remembered framework adapted by changing details
    - Reasoning Operations on Frames
      - **Expectation** — How to select an initial frame to meet some conditions
      - **Elaboration** — How to select and assign sub-frames to represent additional details
      - **Alteration** — How to find a frame to replace one that does not fit well
      - **Novelty** — What to do if no acceptable frame can be found?
      - **Learning** — What frames should be stored or modified as a result of experience?
  - **semantic networks**
    - directed labeled graphs
    - represent concepts & relationships
    - concepts refer to things
- **Knowledge Model (KR Language)**
  - loosely based
  - universe of discourse entities
    - basic types — integers, floats, ..
    - logical constraints — True & False
  - **Class**
    - sets of entities
    - meta class — a class that has another class as an instance
  - **individual**
    - entity that is not a set
    - instance of class
  - **relations**
    - instance-of — class membership
    - type-of — inverse of instance-of
    - subclass-of
  - **Properties (slots)**
    - Own Slots
      - describes direct properties of an entity
      - equivalent to binary relation
      - both classes & individuals can have own slots
    - Template Slots
      - describe properties of instances of a class
      - values are inherited to instances as own slot values
- **constrains & descriptors**
  - **Properties of classes**
    - **Necessary Properties** — must be true
    - **Sufficient Properties** — can conclude class membership
  - **Classes Types**
    - **Primitive Class** — only necessary properties
    - **Defined Class** — sufficient properties defined in terms of another class
  - **Domain** — entity having slot value must be instance of domain class
  - **Range** — slot value must be instance of range class
  - **Facets** — provide more info about slots
    - can be own facet or template facet
- **Inference & Reasoning**
  - Default values
    - generally true but can be overriden
    - problems with default values
      - conflicting values from multiple super classes
      - values override in subclasses
    - methods for conflict resolution
      - Skeptical inheritence — inherit only when no contradiction
      - Override inheritence — more specific class overrides general
      - heuristic — priorities
  - Controlling the inference level
    - Direct — return only directly asserted values
    - Taxonomic — return only inherited values
    - All-inferable — return all possible values
    - Completeness — return true if all values computed with certainty
  - mandatory vs optional operations
    - mandatory — core operations must be supported
    - optional — can be defined using mandatory operations
- **Limitations**
  - difficulty in representing some relations
  - negation cannot be represented uniformly
  - quantification is not part of the language

SPARQL (SPARQL Protocol and RDF Query Language)

- **Introduction**
  - It's a query language for semantic web
  - pull values from RDF data
- **Query Anatomy**
  - Namespace declarations — PREFIX
  - Dataset definition — FROM
  - Result clause — SELECT/ASK/CONSTRUCT
  - Triple patterns — WHERE
  - Modifiers — ORDER BY/LIMIT
- **Architecture & Endpoints**
  - HTTP service for queries — **SPARQL Endpoint**
  - **Generic Endpoint**
    - ARQ
    - Virtuoso
    - Rasqal
  - **Result Formats**
    - XML
    - JSON
    - RDF
    - HTML
- **Language Features**
  - starts with ? — **Variables**
  - Subject-Predicate-Object — **Triple Patterns**
  - Boolean constraints — **FILTER**
  - non-failing matches — **OPTIONAL**
  - **Aggregates**
    - COUNT
    - SUM
    - AVG
    - MIN
    - MAX
  - Federated Queries — **Service**
- **Datasets & Examples**
  - people & relationships — FOAF
  - RDF version of wikipedia — DBPedia
  - creative commons music — Jamendo
  - public agency datasets — UK Government

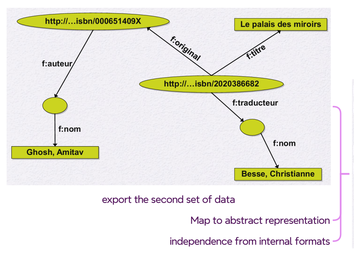
RDFS (Resource Description Framework Schema)

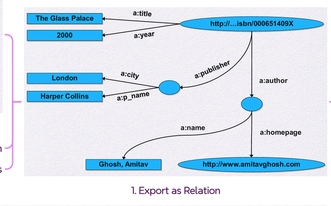
- **Definition & Purpose**
  - RDF Vocabulary Description Language
  - Defines terms & Restrictions
  - Semantic Relationships
  - Typed Hierarchy
- **Core Classes**
  - rdfs:Resource
  - rdfs:Class
  - rdfs:Literal
  - rdfs:Property
  - rdf:statement
  - rdfs:Container
  - rdf:type
- **Core Properties**
  - rdfs:subClassOf
  - rdfs:subPropertyOf
  - rdfs:domain
  - rdfs:range
  - rdfs:label
  - rdfs:comment
  - rdfs:seeAlso
  - rdfs:isDefinedBy
- **Schema Characteristics**
  - Property-centric
  - inferred properties
  - URI-based identification
  - Web publication format (XML)
- **Applications & Vocabularies**
  - **Dublin Core**
    - Title
    - Creator
    - Subject
    - Description
    - Date
  - **Data Integration**
    - Linked Open Data (LOD)
    - DBpedia
    - Relational Database bridges
    - RDFa in XHTML

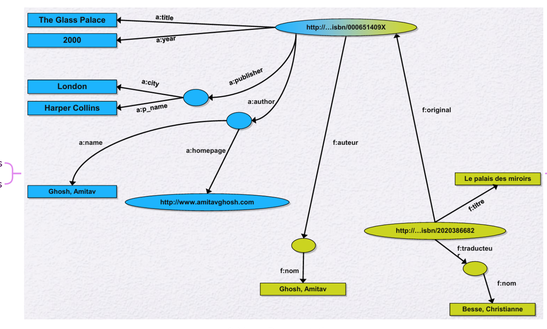
RDF (Resource Description Framework)

- **Core Concepts**
  - statements in RDF can be
    - like this — Property_X — Image
    - or this — Property — Property_i — Statement_i
  - **RDF Triples**
    - ex: http://www.example.org/file.html#home — URI
    - Blank Node
    - Subject
    - property URI
    - Predicate
    - URI
    - literal
    - blank node
    - Object
  - **Resources**
    - web pages
    - XML elements
    - non-web objects
    - identified by URIs
  - **Directed Labeled Graph**
- **Purpose & Benefits**
  - Interoperability
  - machine-understandable information
  - automated processing
  - resource discovery
  - knowledge sharing
  - web of trust
- **Syntax & Serialization**
  - **Formats**
    - RDF/XML
    - Turtle
    - RDFa
  - **XML Styles**
    - serialization syntax
    - abbreviated syntax
  - **namespaces**
    - prefixed for ontologies
    - standardize vocabularies
- **Advanced Features**
  - **Blank Nodes**
    - Internal Identifiers
    - nodeID attribute
    - Merging considerations
  - **Containers**
    - uncordered — Bag
    - ordered — Sequence
    - choice — Alternative
  - Typed Hierarchy — **RDFS (RDF Schemas)**
  - Semantic relations
- **Tools & Querying**
  - Query language — SPARQL
  - Visual browser — Graff

Semantic Web

- **Web of Data**
  - **Current Challenges**
    - Data in isolation (silos)
    - Manual site updates
  - **Required Integration**
    - web scraping/APIs get out of date
    - Standard web technologies
    - Interlinked data
    - Data integration
- **Data Integration Process**
  - 

    export the second set of data
    Map to abstract representation
    independence from internal formats
    1. Export as Relation
  - 

  - 

    Identify shared identities
    Connect different datasets
    2. Merge Representations
  - Ask queries across multiple sets
  - Richer queries with extra glue — 3. Query Integrated Data
- **Technologies & Concepts**
  - **DBpedia**
    - Structures data from wikipedia
    - SPARQL queries
  - Knowledge Enhancements
    - Rule sets
    - Categories
    - graph-based relations
    - machine readability
  - Abstract Formats
- **Applications & Sources**
  - Government data
  - general knowledge
  - company/flight info
  - bioinformatics — scientific data

CommonSense knowledge

- **Definition**
  - basic ability to
    - perceive
    - understand
    - judge
    - shared nearly by all people
  - sources of human reasoning
    - personal experiences
    - world knowledge
- CommonSense Knowledge Graphs (CSKG) & Projects
  - Cyc
  - Open Mind Common Sense
  - Concept Net
  - wiki data
  - visual genome
- **Representation**
  - symbolic
  - NLP
  - Neural
- **Creation Method**
  - like Cyc — top-down — Expert Input
    - important concepts (axioms)
      - dimensions — space — time — scales
      - force — causal connections — material
      - competency vs coverage theories
      - naive physics vs psychology theories
    - limitations
      - brittleness
      - expense of aquisition
      - dependency on language models
    - inductive — knowledge cannot be acquired simply through text
  - CrowdSourcing (from games with a purpose)
  - like Concept Net — bottom up — information extraction
- **Knowledge type**
  - entities & actions
  - inferential & rules
- **Challenges of Common Sense Knowledge**
  - Brittleness of systems
  - expense of acquisition
  - inconsistency in data
  - implicit nature of knowledge

Semantic Networks

- **Quillian Semantic Memory**
  - why?
    - understanding human memory structure
    - used in language understanding
    - encoding dictionary definition of words
    - psychological evidence of associative links
  - **Formalism**
    - Nodes
      - Type Nodes
      - word representation
      - Token Nodes
      - denoting type node in another plane
    - Link Types
      - subclass of
      - disjunctive/conjunctive
      - relates B & C
      - token associated with type node A
      - A modifies B
  - **Meaning of a concept**
    - Dictionary Definition (in a plane)
    - Full Concept — transitive closure of all links — originating from that concept
  - **Reasoning mechanism**
    - Spreading activation
    - unguided breadth first search
- **primitive & Conceptual dependency**
  - **primitives defined as things**
    - interpreter is programmed to understand
  - **Conceptual dependency**
    - reductionist approach
    - canonical form for world knowledge
    - its value — deciding if expressions have the same meaning
  - **five categories**
    - Actions
      - Propel
      - Ingest
      - Ptrans
      - Mtrans
      - Present
    - Tenses
      - Past
      - Future
    - Objects
    - Modifiers of actions
    - Modifiers of objects
- **Structured Inheritence Networks**
  - explicit
    - formal
    - logical meaning for primitives
    - strict definitions
    - classes — Terminology — T-box
    - instances — Assertions — A-box
- **KL-ONE features**
  - Concepts
    - one place logical predicates
  - Roles
    - R(x, y) — two place relation
  - Roles Restrictions
    - class of roles fillers — Value restrictions
    - min/max instances — Number restrictions
  - Meaning
    - necessary conditions only — strict definition or primitive
- **Classification Process**
  - placing new concept under most specific generalizer
- **Semantic Web**
  - URLs as nodes
  - goal of it to retrieve resources based on semantics