Knowledge Representation and Reasoning 22/23

Martim Santos ist195638

1 What is Knowledge?

Declarative vs Procedural
Tacit vs Explicit

Declarative – **knowing that** – knowledge represented as a set of statements.
Ex: Lisbon is the capital of Portugal

Procedural – **knowing how** – knowledge represented as a set of procedures.
Ex: How to ride a bicycle

- Tacit
  - Inconvenient and difficult to formalize and communicate. Not articulated. Person's mental model.
- Explicit
  - Conveyed in formal, systematic representations that are readily communicated.

Nonaka's cycle

- Tacit to Tacit – socialization – we can teach each other by showing rather than speaking.
- Tacit to Explicit – externalization – intensive practices are clarified by putting down on paper.
- Explicit to explicit – combination – putting together different pieces of knowledge.
- Explicit to Tacit - internalization - performing tasks frequently leads to a personal state where we carry out the t. without thinking.

Representation The concept of representation is as philosophically vexing as that of knowledge. Very roughly speaking, representation is a relationship between two domains, where the first is meant to "stand for" or take the place of the second. Usually, the first domain, the representor, is more concrete, immediate, or accessible in some way than the second.

Reasoning In general, it is the formal manipulation of the symbols representing a collection of believed propositions to produce representations of new ones.

Brian Smith calls the ***Knowledge Representation Hypothesis***:

"*Any mechanically embodied intelligent process will be comprised of structural ingredients that a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and b) independent of such external semantic attribution, play a formal but causal and essential role in engendering the behaviour that manifests that knowledge.*"

→ Intelligent processes are comprised of structural ingredients that represent the semantic truth for humans and independently of that play a causal and essential role in engendering the behavior that manifests knowledge.

Two issues: existence of structures that

- we can interpret propositionally.
- determine how the system behaves.

Cognitive penetrability → actions that are conditioned by what is currently believed
Reasoning → Allows the system to infer further information based on what the knowledge available entails (implicitly makes true)
Sound Logic → Generates all answers that are True under common sense
Complete Logic → Generate all answers that are implicit

2 Interpretations, Universals and Existencials

Horn clauses:

- disjunction of literals, where at most one of the literals is positive (i.e., not negated)
    - **Positive/Definite Clause** → exactly one positive literal
- **Negative Clause** → no positive literals

SLD Derivation

- procedure consists of a sequence of steps that successively apply resolution and unification rules until either a solution is found or no more rules apply.
- Remains **undecidable:** infinitely many possible derivations of A from KB, since the rules of resolution and unification can be applied in many different ways and orders.

FOL

In FOL, statements are constructed using variables, constants, predicates, and logical connectives. **Variables** represent objects or values that can be assigned specific values, while **constants** represent fixed objects or values. **Predicates** are expressions that indicate whether a certain relationship holds between objects or values, such as "x is greater than y" or "y is the mother of x". **Logical connectives**, such as "and", "or", and "not", are used to combine statements and form more complex statements. FOL allows for the **quantification of variables**.

∀x [King(x) -> Person(x)] – true for all objects x

∃x [Crown(x) /\ OnHead(x, johnking)] – true for at least one object x

Herbrand Theorem

It states that any sentence in first-order logic that has a model, has a **model that is finite** and **only uses terms built from the constants, functions, and predicates in the original sentence**.

In other words, the theorem provides a way to construct a propositional logic counterpart of any first-order logic sentence, where the propositional logic sentence is satisfiable if and only if the original first-order logic sentence is satisfiable.

A set of clauses *S* is satisfiable if it's Herbrand base is.

- **Herbrand Universe**
  - The Herbrand universe of a set of clauses S in first-order logic (FOL) is the **set of all ground terms that can be constructed using the function and constant symbols in S.**
  - A ground term is a term that contains no variables. The Herbrand universe of a set of clauses S includes all the constants in S, as well as all the function symbols applied to constants in S, recursively.

- the Herbrand universe of S is the set of all terms formed using only the function symbols in S (at least one)

  e.g., if S uses (unary) f, and c, d, $U = \{c, d, f(c), f(d), f(f(c)), f(f(d)), f(f(f(c))), \ldots\}$
- the Herbrand base of S is the set of all $c\theta$ such that $c \in S$ and θ replaces the variables in c by terms from the Herbrand universe

To reason with the Herbrand base it is not necessary to use unifiers, quantifiers and so on, and we have a sound and complete reasoning procedure that is guaranteed to terminate. The catch in this approach is that the **Herbrand base will typically be an infinite set** of propositional clauses. It will, however, be **finite when the Herbrand universe is finite**.

The Herbrand theorem helps address the **tractability problem** by allowing reasoning to be performed on **ground clauses**, **eliminating the complexity associated with reasoning in FOL introduced by variables and quantifiers.** This simplification enables more efficient reasoning and facilitates automated theorem proving.

Resolution

Strategies:

- **Clause Elimination** → Pure clause (contains literal p that does not appear in any other clause) / Tautologies / Subsumed clause (a clause such that one with a subset of its literals is already present)
- **Ordering Strategies** → Unit preference (prefer to resolve unit clauses first)
- **Set of support** → Always use the clauses that come from the query
- **Connection Graph** → build a graph with edges between any two unifiable literals of opposite polarity
- **Special Treatment for Equality**

- **Sorted Logic** → take into account what type a predicate accepts
- **Directional Connectives**

3 Knowledge Based Systems

Definition: A program that:

- Solves **complex problems** in a **particular domain**
- Corresponds to specific tasks
- Leading to the **same solutions a human expert** would find (quality)
- With the **same performance of a human expert** (speed)
- Where the solution is found using intensive knowledge
- **IMPORTANT** to understand that they can only solve **ONE problem** at a time and in **ONE specific domain**.

These systems contain theoretical/public knowledge as well as practical/private knowledge.

Areas where they are good at solving problems:

1. Data interpretation (Dendral, data coming from several kinds of exames ex. X-rays ; Prospector: data interpretation of the geological layers.)
2. Diagnosis (MYCIN: medical diagnosis of bacterial diseases causing serious infections – help the doctor to prescribe appropriate antibiotics)
3. Design (XCOM)
4. Monitoring (Nuclear Plant Monitoring)
5. Planning,
6. Repair (Elf Aquitaine – oil drilling),
7. Scheduling, etc. etc.

Diagnosis → if-then backward chaining

- Certainty factors to choose which rule is fired when many are applicable

Configuration → if-then forward chaining

- No CF needed since alternatives do not need to be checked - ONE solution is enough
- Generate and test algorithm

They use specific knowledge from the experts such as tricks of trade, heuristics that are a product from their hands-on experience.

If-then, rules with higher priority are used.

Success Cases

- Elf-Aquitaine → ocean oil drilling platforms
- MYCIN → written in Lisp; It arose in the laboratory that had created the earlier Dendral expert system -- Was never actually used in practice because: Responsibility issues, Very long interactions

Kinds of Knowledge

Theoretical/public: definitions, facts, classifications, theories, etc about the domain

Practical/Private: tricks of trade, heuristics, specific ways of identifying problem classes and techniques.

About Knowledge

- The emphasis on acquired knowledge based on hands-on experience
- Not automatically learned by the system but introduced in the system
- Extracted from human experts by Knowledge Engineers – complex task – bottleneck
- Requires more than one expert and several knowledge sources, time, effort and expert collaboration
- Requires several iterations
- Represented in a formal language
- KB + inference engine

Problems:

- Do **not recognize their own limitations** or **cases to which they do not apply** or for which existing knowledge is not applicable.
- They have **no independent way to verify their conclusions** and suggestions.
- **Moral question** of **who is responsible for decisions** made

4 Rules in Production Systems

A conditional like P ⇒ Q can be understood as transforming:

- assertions of P to assertions of Q -- (assert P) → (assert Q)
- goals of Q to goals of P -- (goal P) → (goal Q)

Distinguish between

1. **goal vs. data directed reasoning**
   a. goal: from Q towards P
   b. data: from P towards Q
2. **forward vs. backward-chaining**
   a. forward: along the →
   b. backward: against the →

Production systems

- Idea: **working memory** + **production rule set**
- Working memory: like DB, but volatile
- Production rule:
  - **IF** conditions **THEN** actions
    - condition: tests on WM
    - action: changes to WM

Actions → can be **ADD** pattern, **REMOVE** index or **MODIFY** index (attr spec)

Basic operation: cycle of

1. **recognize -** find conflict set: rules whose conditions are satisfied by current working memory
2. **resolve -** determine which rule will fire
3. **act -** perform required changes to WM

Stop when no rules fire.

Example:

Placing bricks in order of size

largest in place 1, next in place 2, etc.

Initial working memory

```
(counter  index 1)
(brick  name A  size 10  place heap)
(brick  name B  size 30  place heap)
(brick  name C  size 20  place heap)
```

Production rules:

```
IF   (brick  place heap  name n  size s)
     -(brick  place heap  size {> s})
     -(brick  place hand)
THEN  MODIFY 1  (place hand)
```

put the largest brick in your hand

```
IF   (brick  place hand)  (counter index i)
THEN  MODIFY 1  (place i)
      MODIFY 2  (index [i+1])
```

put a brick in your hand at the next spot

Applications:

1. Psychological Modeling
2. Expert Systems

Advantages:

- Modularity - each rule acts independently of the others
- Fine-Grained Control - no complex goal or control stack
- Transparency - can recast rules in English to provide explanation of behavior

Conflict Resolution in Production Systems

- Sometimes with data-directed reasoning, we want to fire all applicable rules
- With goal-directed reasoning, we may want a single rule to fire
  - Arbitrary
  - First rule in order of presentation (as in Prolog)
  - Specificity
  - Recency
    - Fire rule that uses most recent WME
    - Fire on least recently used ruke
  - Refractoriness

- Never use same rule for same value of variables (called rule instance)
- only use a rule/WME pair once (will need a "refresh" otherwise)

Conflict Combinations:

- **OPS5** - Discard already used rule instances. Order remaining rule instances in terms of recency of conditions matching. If there is no single rule, order based on the number of conditions. Lastly, choose arbitrarily.
- **SOAR** - System that attempts to find a way to move from a start state to a goal state by applying productions. If unable to decide, it sets up the selection as its new goal.
- **RETE Procedure** (explained below)

RETE Procedure

Early systems spent 90% of their time matching, even with indexing and hashing.

But:

- WM is modified only slightly on each cycle
- Many rules share conditions

So:

- **pass WME through network of tests**
- **tokens that make it through satisfy all conditions and produce conflict set**
- can calculate **new conflict set in terms of old one** and change to WM

→ RETE algorithm provides a way to **avoid repeatedly testing the same conditions** against the working memory by constructing a **network of nodes** that can **quickly identify the rules that are potentially applicable** to the current state of the working memory. This results in significant **performance improvements** over naive production systems that repeatedly test each condition against the working memory.

5 Object-Oriented Representation

Knowledge is represented as **classes** and **instances**. **Reasoning** is based on **inheritance**, **rules** and **procedural attachments** (demons and servants).

There are 2 types of frames :

- **Individual Frames** - represent a single object as a named list of buckets called slots that are completed with fillers.
- **Generic Frames** - represent categories of objects

Frames Systems are a family of KR systems based on an **Object Oriented representation** of knowledge. Proposed by Marvin Minsky.

The main idea is that when we have some **pre-existing idea** of anything we already know what to expect and what characterizes what we want to talk about.

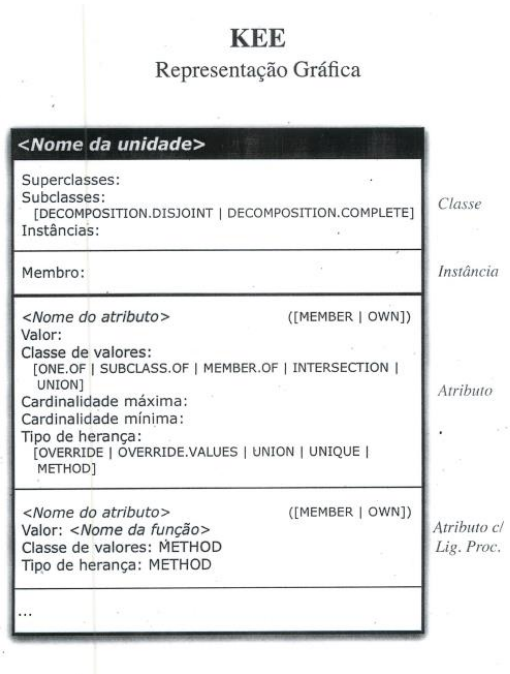
Concepts are **bundled together and encapsulated** in the object that represents all we know about that concept.

Representation

- All that is known about a particular entity is encapsulated in an object, representing that entity.
- Objects are organized hierarchically
- Connected to more general classes and to more specific ones.
- A **Frame** is an object connected with others (or not). It can represent **classes or instances**. Frames are characterized by attributes. These attributes are propagated down the hierarchy. The **attributes** are called **Slots**. Slots have values.

Reasoning

- **Inheritance** of attributes/properties/slots – basic reasoning of mechanism of Frames that all systems must provide. Slots are automatically propagated down. Some systems only allow single inheritance while others allow multiple.
- **Procedural attachment** – attaching procedural knowledge with declarative knowledge. Pieces of code that calculate the value of a particular slot.
  Two types of procedural attachment:
    - Servants: fired when we want to know the value of the slots. Ex: age
    - Daemons: fired when we change the value in the slot.
- **Rule** based systems: These are normal systems based on rules, using a production system approach.



KEE stands for **Knowledge Engineering Environment** and was a tool to develop Expert Systems. The basis for the representation of knowledge was a **Frame System** which is going to be explained in the following. Had a Belief Revision system that ensured that contradictions in the systems would be reported to the knowledge engineer introducing knowledge, another that would allow the system to perform inference in a given context, giving us the possibility to analyse the consequences of a set of assertions, among many others.

Representation

- A KEE KB is composed of one or more hierarchies of Frames/Units
- **Classes and instances**
- **Classes**: **Superclass**, **Subclasses** and **instances** directly connected to the Frame.
- A class is allowed to have more than one parent → **multiple inheritance**
- Instances are described by the class to which they belong to.
- Speed-up: add info about subclasses in the superclass
- Two types of **slots**:
    - Member-slot: attribute that all members of the class have
    - Own slot: attribute that pertain only to the unit where the slot is attached to.
- All member slots when inherited by a class continue to be a member slot. When they reach an instance, they transform to own slots.
- Facets that characterize slots are: value, class of values, minimum and maximum cardinality, and kind of inheritance.
- More:
    - Use NIL to say it does not have.
    - Use "–" if it is unknown.
    - All possibilities are allowed as value of a slot: Classes, Instances and access to all LISP data types (lists, atoms, strings, all sorts of numbers, hash-tables, etc.)
    - Facet Max/Min Cardinality: specifies the maximum and minimum number of values that the slot may have. It is a positive integer.
    - Facet Inheritance: describes how the value of the slot is going to be propagated to the units down in the hierarchy.
        - OVERRRIDE – If an own value is introduced it is the value of the slot because it stops/prevents the propagation of the value inherited from the parent units.
        - OVERRRIDE.VALUES – Similar to the previous type but for attributes that have a maximum cardinality higher than 1.
        - UNION – Union of all inherited values from the parents plus the eventual value directly introduced.
        - UNIQUE – Blocks the inheritance of the value of the slot.
        - METHOD – to specify procedural attachment.

Reasoning

- **Inheritance** - of slots along the hierarchies of units
- **Procedural attachment** – Lisp code fires when unit it being questioned

```
(defun quantity-of-food (unit)
        (/ (*   (get-value unit 'weight)
                (get-value unit 'height))
        (get-value unit 'age)))
```

- **Rules** - if then rules
    - can be written in a Lisp or human like language
- KEE allowed both forward reasoning through the ASSERT command and backward reasoning through the QUERY command.

6 Frames - Analysis Strengths Weaknesses

What are the **main advantages** of representing knowledge using Frames vs Logic?

- **All knowledge** about a given concept is **grouped** in its corresponding frame. Combines **declarative and procedural knowledge.** For Humans, it is **natural** to represent knowledge in classes and instances with attributes
- Inheritance is a natural form of reasoning

What are we **missing** in Frames **(Weaknesses**)?

- FOL components: Language, Syntactic System, Semantic System
- Frame systems, in general, **lack a formal semantics**
- **Expressive Power**?
    - Disjunctions?

Knowledge Level vs Symbol Level

Allen Newell's analysis (KRR Book):

- Knowledge level: deals with language, entailment
- Symbol level: deals with representation, inference

Picking a logic has issues at each level

- Knowledge level: expressive adequacy, theoretical complexity, ...
- Symbol level: architectures, data structures, algorithmic complexity

7 Description Logics

They focus on the **definition of concepts** (no exceptions). Start from the basics (small is beautiful philosophy) and keep adding knowledge representation primitives (expressiveness layers) while ensuring we stay within the decidability barrier.

Motivation -> Combine frame's natural representation and reasoning primitives with the semantics in FOL. They solved the **Semi-decidability of FOL** and **lack of semantics of frames.**

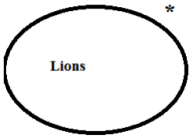
Basic Ideas DL's

- **Small is beautiful**.
  - Development by layers
  - Formalism is developed and named.
- Decidability
  - Development by layers
    - It needs to be tractable, efficient.

KL-ONE [Brachman]

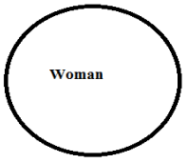
A **very efficient KRR system**, developed by layers, with the small is beautiful philosophy

- 2 Components:
  - Terminological component
    - **T-BOX** – definitions of **concepts**, where terminology is defined
  - Assertional component
    - **A-BOX** – describes a particular world, the assertions and **instances** a particular word
- **Concepts**
  - Generic concepts – represent classes
    - Defined – "totally" defined; necessary and sufficient conditions (NO *)
      - Woman is a Female Human
    - Primitive – partially defined; defined through necessary conditions (*)
      - Lions roar

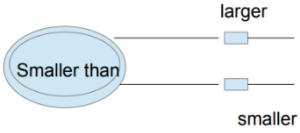


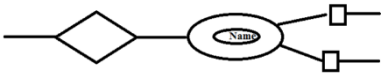
Taxonomic relations – used to define a **hierarchy of concepts.**

- All concepts are related to at least another concept.
  - TOP MOST concept = Thing



- **Roles**: a generalization of attribute, property, constituent, part-of, etc
  - Restricting – Class of values and cardinalities (**restr**)
  - Differentiating a version of an inherited role (**diff**)
- **Structural conditions**
  - They introduce conditions/restrictions that must be satisfied by roles or by the values of roles.
    Some structural conditions often used:





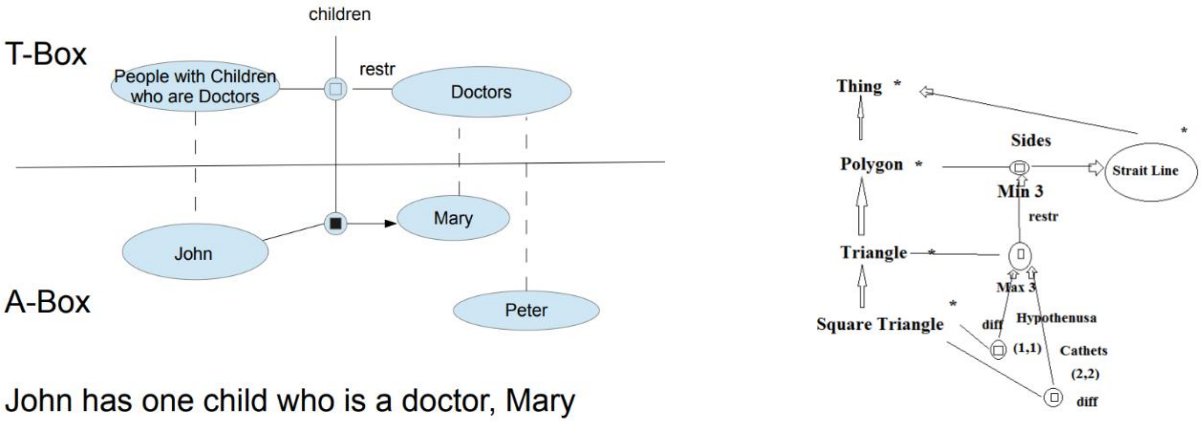


Image: KL-ONE Examples

To define a concept

- Reasoning
  - Inheritance
  - Subsumption
  - Classification

Semantics

- Each concept represents a set of entities – a subset of D.
- Each role represents a set of pairs of these entities - a subset of DxD
- Both concepts and roles correspond to relationships in FOL
- In DL's we associate to **each concept** and to **each role** the **relations** denoted by them – these associated sets are called **Extensions**.

AL+

- Three types of non-logical symbols:
  - atomic concepts (Dog, Teenager, Thing)
  - roles (:Age, :Parent)
  - constants (johnSmith)
- Four types of logical symbols:
  - punctuation: [,], (,)
  - positive integers: 1, 2, 3, ...
  - concept-forming operators: ALL, EXISTS, FILLS, AND
  - connectives: =, !=, and →

A DL knowledge base is a set of DL sentences serving mainly to

- give names to definitions
  - e.g. (FatherOfDaughters = [AND Male [EXISTS 1 :Child] [ALL :Child Female]])
- give names to partial definitions
  - e.g. (Dog = [AND Mammal Pet CarnivorousAnimal [FILLS :VoiceCall barking]])
- assert properties of individuals
  - e.g. (joe → [AND FatherOfDaughters Surgeon]])

Normalization

Repeatedly apply the following operations to the two concepts:

- expand a definition:
  - replace an atomic concept by its KB definition
- flatten an AND concept:
  - [AND ... [AND d e f] ...] ⇒ [AND ... d e f ...]
- combine the ALL operations with the same role:
  - [AND ... [ALL r d] ... [ALL r e] ...] ⇒ [AND ... [ALL r [AND d e]] ...]
- combine the EXISTS operations with the same role:
  - [AND ... [EXISTS n1 r] ... [EXISTS n2 r] ...] ⇒ [AND ... [EXISTS n r] ...] (where n =Max(n1 ,n2))
- remove a vacuous concept:
  - Thing, [ALL r Thing], [AND]
- remove a duplicate expression

```
[AND    Person
        [ALL :Friend Doctor]
        [EXISTS 1 :Accountant]
        [ALL :Accountant [EXISTS 1 :Degree]]
        [ALL :Friend Rich]
        [ALL :Accountant [AND Lawyer [EXISTS 2 :Degree]]]]
```

⇩

```
[AND    Person
        [EXISTS 1 :Accountant]
        [ALL :Friend [AND Rich Doctor]]
        [ALL :Accountant [AND Lawyer [EXISTS 1 :Degree] [EXISTS 2 :Degree]]]]
```

⇩

```
[AND    Person
        [EXISTS 1 :Accountant]
        [ALL :Friend [AND Rich Doctor]]
        [ALL :Accountant [AND Lawyer [EXISTS 2 :Degree]]]]
```

9 Semantic Web

A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. By Tim Berners-Lee, James Hendler and Ora Lassila

- The Semantic Web will enable machines to **comprehend** semantic documents+data, not human speech and writings.
  - Expressing Meaning
  - Knowledge Representation
  - Ontologies
  - Agents
- **Evolution** of Knowledge
  - Elaborate, Precise Searches

- Properly designed, the Semantic Web can assist the evolution of human knowledge as a whole.

The Vision

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web.

Ontology – describe the concepts of a domain

Logic + Proof = Knowledge Representation Language

- Will build on **XML**'s ability to define customized tagging schemes and **RDF**'s flexible approach to representing data

RDF

- RDF is a **datamodel** for **objects** ("resources") and **relations** between them, provides a **simple semantics** for this datamodel, and these datamodels can be **represented in an XML syntax**.
- **RDF Schema** is a vocabulary for **describing properties and classes** of RDF resources, with a semantics for **generalization-hierarchies** of such properties and classes.

OWL

- The first level above RDF required for the Semantic Web is an **ontology language** that can **formally describe the meaning of terminology used in Web documents**.
- If machines are expected to perform useful reasoning tasks on these documents, the language must go **beyond the basic semantics of RDF Schema**.
- OWL **adds more vocabulary** for describing properties and classes:
  - Relations between classes (e.g. disjointness)
  - Cardinality (e.g. "exactly one"),
  - Equality,
  - Richer typing of properties,
  - Characteristics of properties (e.g. symmetry),
  - Enumerated classes.
  - Etc.

The layers:

- **OWL Lite** – supports users primarily needing classification hierarchy and simples constrains.
- **OWL DL** - supports users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- **OWL Full** - for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

10 Inheritance

Inferential distance → not linear distance, but topologically based: node a is closer to node b than to node c if there is a path from a to c through b.

Idea: conclusions from b preempt those from c

Different forms of reasoning:

- **credulous reasoning**: Choose a preferred extension, perhaps arbitrarily, and believe all of the conclusions supported by it.
- **skeptical reasoning**: Believe the conclusions supported by any path that is present in all preferred extensions.
- **ideally skeptical reasoning**: Believe the conclusions that are supported by all preferred extensions. This is subtly different from skeptical reasoning, in that these conclusions may be supported by different paths in each extension. One significant consequence of this is that ideally skeptical reasoning cannot be computed in a path-based way.

Credulous extension → Any plausible set of conclusions

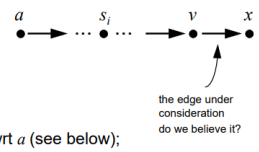
Support and admissibility

Γ supports a path $a \cdot s_1 \cdot \ldots \cdot s_n \cdot (\neg)x$ if the corresponding set of edges $\{a \cdot s_1,\ \ldots,\ s_n \cdot (\neg)x\}$ is in E, and the path is admissible according to specificity (see below).

the hierarchy supports a conclusion $a \rightarrow x$ (or $a \not\mapsto x$) if it supports some corresponding path

A path is admissible if every edge in it is admissible.

An edge $v \cdot x$ is admissible in Γ wrt a if there is a positive path $a \cdot s_1 \ldots s_n \cdot v$ $(n \geq 0)$ in E and



1. each edge in $a \cdot s_1 \ldots s_n \cdot v$ is admissible in Γ wrt a (recursively);
2. no edge in $a \cdot s_1 \ldots s_n \cdot v$ is redundant in Γ wrt a (see below);
3. no intermediate node $a, s_1, \ldots, s_n$ is a preemptor of $v \cdot x$ wrt a (see below).

A negative edge $v \cdot \neg x$ is handled analogously.

Credulous extensions

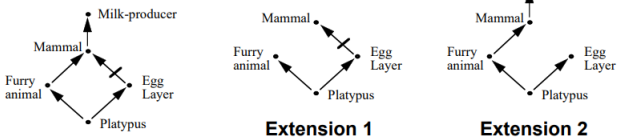
Γ is a-connected iff for every node x in Γ, there is a path from a to x, and for every edge $v \cdot (\neg)x$ in Γ, there is a *positive* path from a to v.

In other words, every node and edge is reachable from a

Γ is (potentially) ambiguous wrt a node a if there is some node $x \in V$ such that both $a \cdot s_1 \ldots s_n \cdot x$ and $a \cdot t_1 \ldots t_m \cdot \neg x$ are paths in Γ

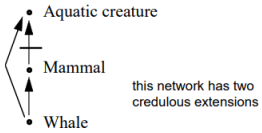
A credulous extension of Γ wrt node a is a maximal unambiguous a-connected subhierarchy of Γ wrt a

If X is a credulous extension of Γ, then adding an edge of Γ to X makes X either ambiguous or not a-connected



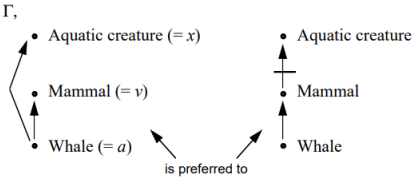
Preferred extensions

Credulous extensions do not incorporate any notion of admissibility or preemption.



Let X and Y be credulous extensions of Γ wrt node a. X is preferred to Y iff there are nodes v and x such that:

- X and Y agree on all edges whose endpoints precede v topologically,
- there is an edge $v \cdot x$ (or $v \cdot \neg x$) that is *inadmissible* in Γ,
- this edge is in Y, but not in X.



A credulous extension is a preferred extension if there is no other extension that is preferred to it.

11 Knowledge Sharing and Ontologies

Knowledge Sharing

- Problem:
  - The cost of knowledge based systems
  - Building the knowledge base from scratch
- KB Components
  - Medical diagnosis and medical tutoring
    - Vocabulary definition: disease, organ, bacteria – **ontologies**
  - Electronic diagnosis vs medical diagnosis
    - Raise hypothesis, test, refine, etc – **problem solving methods**

Solution:

- Reuse and Sharing of knowledge
  - Translation of knowledge bases between different KR languages
  - Arbitrary differences among systems belonging to the same family
  - Remote access to the knowledge base of another system
  - Meaning of what is shared: Lack of consensus about vocabulary

What is an ontology?

- Capture the static knowledge in a given domain that is accepted and sharable across applications and groups
- Defs:
  - An **explicit formal specification** of a **shared conceptualization**
  - a vocabulary of terms and some specification of their meaning

- Distinction ontology/KB
  - Different role played by represented knowledge
    - Ontologies - +/- consensual of a community
    - Knowledge specific of a domain
- Depend on the application that powered its construction
  - Same domain, different tasks

Kinds of ontologies

- **Representation** or **meta-**
  - capture the **representation primitives** in a KR family or paradigm (Frames: class, instance, relation -slots and facet-, function, etc.)
- **General** or **upper**
  - capture very general notions applicable across domains (Time: time-point, time-range, duration, overlaps, before, after, etc.)
- **Domain**
  - specific of a particular domain (Chemical elements: elements, non-reactive elements, helium, non-metals, carbon, etc.)

Origin

- An Ontology that describes the processes of the Central Dogma of Molecular Biology for prokaryotic organisms.
- Formalized and can be used by an inference engine to answer user questions.

How are they built?

- General process
  - Specification
  - Conceptualization
  - Formalization
  - Implementation
  - Maintenance
  - Knowledge Acquisition
  - Evaluation
  - Documentation
- Life Cycle
  - **Iterative** (end of the iteration, update)
  - **Downfall** (keeps moving forward, never backwards)
  - **Evolutive** (update at each step)
- Build from scratch
  - Few methodologies, none is widely accepted.
  - Most representative methodologies are:
    - TOVE methodology
      - Competency questions, the most used technique for requirement specification
    - ENTERPRISE methodology
      - Middle-out approach for conceptualization/formalization
    - METHONTOLOGY
      - Intermediate tabular representations for formalization

Techniques

- Knowledge acquisition:
  - brainstorming, interviews, questionnaires, text analysis, mind maps
  - experts, books, norms, etc.
- Conceptualization
  - Middle-out, grouping, glossary of terms, concept classification trees
- Formalization
  - intermediate tabular representations (concept dictionary, table of binary relations, etc.)

Ontology evaluation

- One needs to guarantee quality
  - **technical evaluation**: judge ontologies, their software environment and documentation against a framework:
    - consistency, completeness, conciseness, etc.

- **user assessment**: judge from the user point of view the usability and usefulness of ontologies, their software environment and documentation when they are reused or shared in applications
  - understandability, technically evaluated, portable, etc.
- ONTOCLEAN: analyze hierarchical taxonomy using philosophical principles.
  - Aims:
    - assure that instances do not violate class properties
    - assure consistent hierarchical structure

12 Reiter's Default Logic

Closed-world Assumption -> principle used in knowledge representation that assumes that everything that is not explicitly known to be true is false.

Default logic is a non-monotonic logic proposed by Raymond **Reiter** to formalize **reasoning with default assumptions**.

Default logic can express facts like "**by default, something is true**"; by contrast, standard logic can only express that something is true or that something is false.

A logic is **non-monotonic** if some conclusions can be **invalidated by adding more knowledge**. The logic of definite clauses with negation as failure is non-monotonic. Non-monotonic reasoning is useful for **representing defaults**. A default is a rule that can be used unless it overridden by an exception.

A default theory is a pair **<D, W> (ψ, Δ)**. **W (or Δ)** is a set of logical formulas (wff's in FOL), called the *background theory*, that formalize the facts that are known for sure. **D (or ψ)** is a set of default rules, each one being of the form:

$$\frac{\text{Prerequisite} : \text{Justification}_1, \ldots, \text{Justification}_n}{\text{Conclusion}}$$

The default rule "birds typically fly" is formalized by the following default:

$$D = \left\{ \frac{\text{Bird}(X) : \text{Flies}(X)}{\text{Flies}(X)} \right\}$$

This rule means that, "if X is a bird, and it can be assumed that it flies, then we can conclude that it flies".

$$W = \{\text{Bird}(\text{Condor}), \text{Bird}(\text{Penguin}), \neg\text{Flies}(\text{Penguin}), \text{Flies}(\text{Bee})\}.$$

According to this default rule, a condor flies because the precondition *Bird(Condor)* is true and the justification *Flies(Condor)* is not inconsistent with what is currently known. On the contrary, *Bird(Penguin)* does not allow concluding *Flies(Penguin)*: even if the precondition of the default *Bird(Penguin)* is true, the justification *Flies(Penguin)* is inconsistent with what is known. From this background theory and this default, *Bird(Bee)* cannot be concluded because the default rule only allows deriving *Flies(X)* from *Bird(X)*, but not vice versa. Deriving the antecedents of

an inference rule from the consequences is a form of explanation of the consequences, and is the aim of abductive reasoning.

How do we compute the extension

- (Ψ, Δ)
  - $\Delta = \{Q(nixon), R(nixon)\}$
  - $\Psi = \{Q(x):P(x)/P(x);\ R(x):\neg P(x)/\neg P(x)\}$
- Close theory
  - Pick all constants and close all default rules for those instances
- Make proposals involving the closed default rules
- Verify ALL proposal extensions

Extension proposals

- Prop1 = Th(Δ)
- Prop2 = Th(Δ U {P(nixon)})
- Prop3 = Th(Δ U {$\neg$P(nixon)})
- NOT
  - Prop4 = Th(Δ U {P(nixon), $\neg$P(nixon)})
  -
  - The contradictions in the extensions can only arrise if they are already in Δ

Verify all extensions

- Usually
  - Starting from those with more information
- $\Delta = \{Q(nixon), R(nixon)\}$
- $\Psi = \{Q(nixon):P(nixon)/P(nixon);\ R(nixon):\neg P(nixon)/\neg P(nixon)\}$
- Prop1 = Th(Δ)
- Prop2 = Th(Δ U {P(nixon)})
- Prop3 = Th(Δ U {$\neg$P(nixon)})

Verification

- Prop1 = Th(Δ)
- E0 = {Q(nixon), R(nixon)} Ψ_2
- E1=Th({Q(nixon), R(nixon)}) U {$\neg$P(nixon), P(nixon)} $\Psi 1$
- E2= it will not converge to the proposal !!!
  - Once k enters the extension it will not leave

Verification

- Prop3 = Th(Δ U {$\neg$P(nixon)})
- E0 = {Q(nixon), R(nixon)} Ψ_2
- E1=Th({Q(nixon), R(nixon)}) U {$\neg$P(nixon)}
- E2=Th({Q(nixon), R(nixon), $\neg$P(nixon)}) U {$\neg$P(nixon)}
- E3=E2
- So the process has converged to a fixed point which is Prop3

Verification

- Prop2 = Th(Δ U {P(nixon)})
- E0 = {Q(nixon), R(nixon)} $\Psi 1$
- E1=Th({Q(nixon), R(nixon)}) U {P(nixon)}
- E2=Th({Q(nixon), R(nixon), P(nixon)}) U {P(nixon)}
- E3=E2
- So the process has converged to a fixed point which is Prop2

Important results

- Reiter identified a subclass of **closed default theories** that are guaranteed to have extensions
- **Normal Default theory** is a theory (Y, D) where all default rules are normal, A(x):C(x)/C(x)

THEOREM: **All Normal Default theories have an extension**

- **Semi-monotonicity Theorem**
    - Let ψ and ψ´ two sets of closed normal default theories and ψ´ ≤ ψ. Let E´ be the extension of the closed normal default theory (ψ´, Δ). Then (ψ, Δ) has an extension E such that E´ ≤ E.

    So, normal default theories are well behaved but sometimes they are not enough to get us the results we want.

- **Semi-normal rules** look like:
    - A(x):((C(x)^D(x))/C(x)
    - They are not guaranteed to have an extension BUT they **follow the semi-monotonicity property**.

(In Portuguese)

Definição de extensão de uma teoria de omissão: Ω é uma extensão da teoria de omissão (Ψ, Δ) sse $\Gamma(\Omega) = \Omega$, em que $\Gamma(\Omega)$ é o menor conjunto que satisfaz as seguintes condições:

1. $\Delta \subset \Gamma(\Omega)$
2. $\Gamma(\Omega)$ é fechado quanto à derivabilidade, isto é, $th(\Gamma(\Omega)) = \Gamma(\Omega)$
3. Se $\frac{\alpha : \beta}{\gamma} \in \Psi$, $\alpha \in \Gamma(\Omega)$ e $\neg\beta \notin \Omega$ então $\gamma \in \Gamma(\Omega)$

Assim, para Ω ser uma extensão da teoria de omissão (Ψ, Δ), tem que satisfazer as cinco condições enunciadas na definição acima: Ω é um ponto fixo do operador Γ, $\Gamma(\Omega)$ é mínimo, e as condições 1. a 3. são satisfeitas.

Método para verificar se um conjunto de fbfs é extensão de uma teoria de omissão: Seja $\epsilon \subseteq \mathcal{L}$ um conjunto de fbfs fechadas e seja (Ψ, Δ) uma teoria de omissão fechada. Defina-se:

$$\epsilon_0 = \Delta$$

$$\epsilon_{i+1} = th(\epsilon_i) \cup \{\gamma : \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \in \Psi, \text{ em que } \alpha \in \epsilon_i \text{ e } \neg\beta_1, \ldots, \neg\beta_n \notin \epsilon\}$$

então, ϵ é uma extensão de (Ψ, Δ) ssè $\epsilon = \bigcup_{i=0}^{\infty} \epsilon_i$.

Nota: Este não é um método construtivo por causa do ϵ que está a bold na definição, o que implica que se tem que ter à partida uma proposta de extensão.

Extra information

AL+

Language:

- **Expressiveness**: AL+ includes additional features such as sets of individuals ([ALL :Child [ONE-OF wally theodore]]), role hierarchies, and qualified number restrictions ([AND [EXISTS 2 :Child] [ALL :Child Female]]). This increased expressiveness. AL+ does not support cyclic definitions or the representation of contextual information.
- **No Defaults**
- **Monotonicity**: AL+ is monotonic, meaning that new information can only be added to the knowledge base without contradicting previously known information.
- **Well-defined semantics**
- **Natural and intuitive representation of knowledge**

Reasoning:

- **Decidability**: AL+ is decidable, meaning that automated reasoning tasks can be performed algorithmically with finite resources.
- **Inheritance**: Inheritance is a central concept in AL+ and is used to derive new knowledge from existing knowledge in the knowledge base.
- **Classification**: AL+ supports the classification of individuals into classes
- **Subsumption**: Subsumption testing is an important reasoning task in AL+ and is used to determine whether one class is a subset of another.

KL-ONE

Language:

- **Expressiveness**: KL-ONE supports inheritance and does not have support for role hierarchies or qualified number restrictions. However, it does support description composition and has a rich set of constructs for specifying attributes and relations among classes.
- **No Defaults**
- **Monotonic**
- **Well-defined formal semantics**
- **Natural and intuitive representation of knowledge**

Reasoning:

- **Decidability**: AL+ is decidable, meaning that automated reasoning tasks can be performed algorithmically with finite resources.
- **Inheritance**
- **Subsumption –** see if one concept is more general than another
- **Classification –** find the place in the hierarchy where the concept should be introduced

Reiter's Default Logic

Language:

- **Expressiveness**: Allows for the representation of complex default rules that specify how default values should be inferred for classes and individuals.
- **Defaults**: Reiter's default logic is built specifically to handle default reasoning and has explicit support for default rules that can be used to infer default values for classes and individuals.
- **Monotonicity**: Reiter's default logic is non-monotonic, meaning that new information can lead to the retraction of previously inferred conclusions. This is due to the way that defaults work in Reiter's logic.
- **Well-defined formal semantics**
- **Natural representation**: Reiter's default logic can be used to represent knowledge, particularly when dealing with defaults and uncertain information.

Reasoning:

- **Decidability**: Reasoning in Reiter's default logic is semi-decidable, meaning that it is possible to check whether a conclusion can be inferred from a set of default rules, but it may not always be possible to do so in a finite amount of time.
- **Rules/Default Reasoning and Inference**