Knowledge Representation

Philipp Koehn

7 March 2024

Outline

- Representation systems
- Categories and objects
- Frames
- Events and scripts
- Practical examples
  - Cyc
  - Semantic web

representation systems

Knowledge

- Goal: common sense reasoning
- Need to represent knowledge about the world
- Types of knowledge
  - objects
  - events
  - procedures
  - relations
  - mental states
  - meta knowledge

Properties of Representation Systems

- Representational adequacy
  - ability to represent the required knowledge
- Inferential adequacy
  - ability to manipulate knowledge

  ⇒ produce new knowledge
- Inferential efficiency
  - ability to direct inference methods into productive directions
  - ability to respond with limited resources (time, storage)
- Acquisitional efficiency
  - ability to acquire new knowledge
  - ideally, automatically

categories and objects

Categories

- Specific **objects**, e.g., my basketball $\texttt{BB}_9$

- General **category**, e.g., `Basketballs`
  - categories as relationships: $\texttt{Basketballs(BB}_9\texttt{)}$
  - reification of predicate: `Basketballs`
    → use in other predicates $\texttt{Member(BB}_9\texttt{, Basketballs)}$
    abbreviated to $\texttt{BB}_9 \in \texttt{Basketballs}$

- **Subcategories**
  - for instance `Subset(Basketballs, Ball)`
  - abbreviated as $\texttt{Basketballs} \subset \texttt{Ball}$

- **Taxonomy**: System of categories and subcategories

Basic Relations for Categories

- `Disjoint({Animals, Vegetables})`
- `ExhaustiveDecomposition(`
  `{Americans, Canadians, Mexicans},`
  `NorthAmericans)`
- `Partition({Males, Females}, Animals)`

- These properties can be defined with first order logic

Physical Composition

- Basic relations such as `PartOf`
  - `PartOf(Bucharest, Romania)`
  - `PartOf(Romania, EasternEurope)`
  - `PartOf(EasternEurope, Europe)`
  - `PartOf(Europe, Earth)`
- Can be used to define **composite objects**

$$\begin{aligned}\texttt{Biped}(a) \Rightarrow \exists l_1, l_2, b\, \texttt{Leg}(l_1) \wedge \texttt{Leg}(l_2) \wedge \texttt{Body}(b) \\ \wedge\, \texttt{PartOf}(l_1, a) \wedge \texttt{PartOf}(l_2, a) \wedge \texttt{PartOf}(b, a) \\ \wedge\, \texttt{Attached}(l_1, b) \wedge \texttt{Attached}(l_2, b) \\ \wedge\, l_1 \neq l_2 \\ \wedge \left[\forall l_3 \texttt{Leg}(l_3) \wedge \texttt{PartOf}(l_3, a) \Rightarrow (l_3 = l_1 \vee l_3 = l_2)\right]\end{aligned}$$

Prototypes

- Recall: natural categories are hard to define
- There is no set of features that applies to all instances
- But: prototypes have such properties

- Select **typical** members of categories

$$\exists b \in \texttt{Typical(Bird)} \Rightarrow \texttt{CanFly}(b)$$

Hierarchies and Inheritance

- Hierarchy (or taxonomy) is a natural way to structure categories
- Importance of abstraction in remembering and reasoning
  - groups of things share properties in the world
  - we do not have to repeat definitions
- Example: saying "elephants are mammals" is sufficient to know a lot about them
- Inheritance is the result of reasoning over paths in a hierarchy:

  "does a inherit from b?"

  is the same as

  "is b in the transitive closure of :IS-A (or subsumption) from a?"

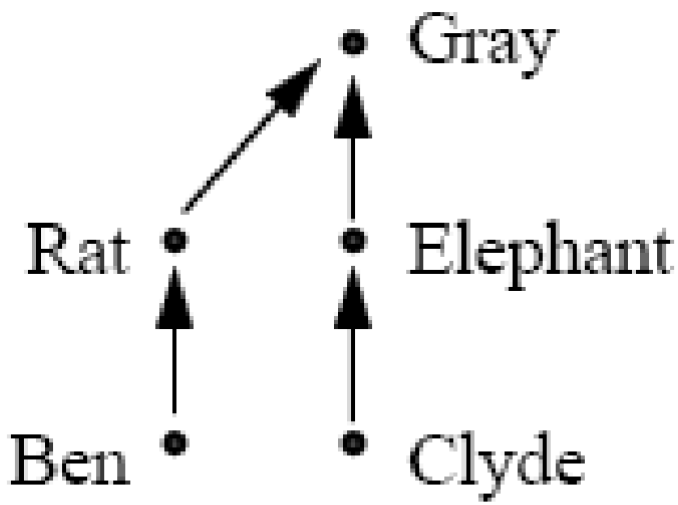
Graphical Representation of Inheritance

- IS relations:

$$\begin{array}{c}\text{Clyde}\\ \downarrow \\ \text{Elephant (category)}\\ \downarrow \\ \text{Gray (property)}\end{array}$$

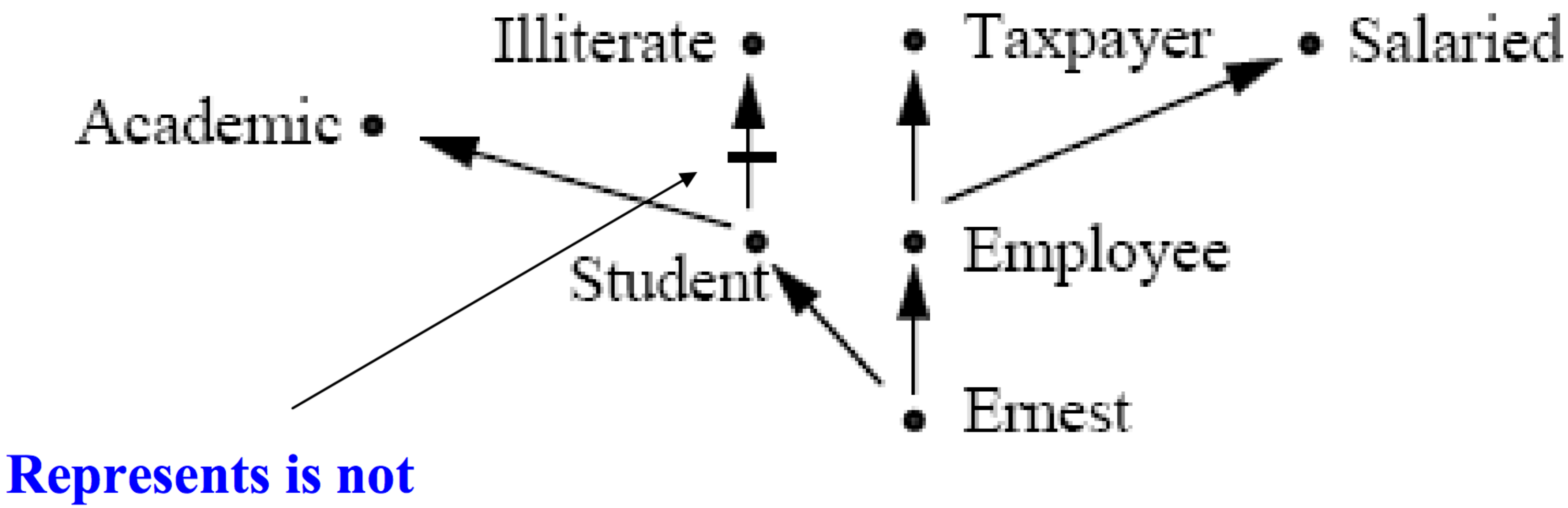
- Clyde is an Elephant, Elephant is Gray
- Reasoning with paths and conclusions they represent ("Transitive relations")
- Transitive closure
  Clyde is Elephant, Elephant is Gray $\Rightarrow$ Clyde is Gray

Strict Inheritance



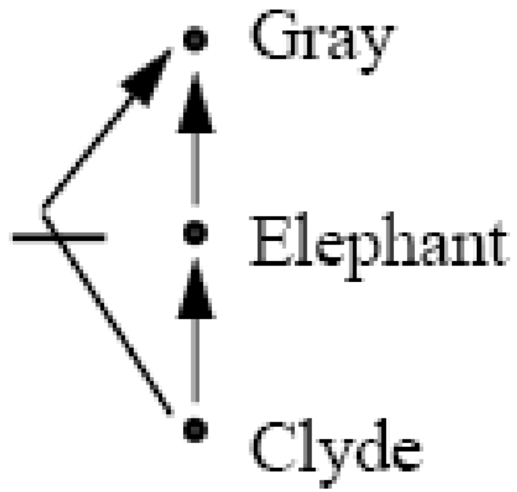
- Conclusions produced by complete transitive closure on all paths (any traversal procedure will do)
- All reachable nodes are implied

Lattice Structure with Strict Inheritance



- Multiple AND (∧) parents (= multiple inheritance)
- Trees: all conclusions you can reach by any paths are supported

Defeasible Inheritance



Elephants are gray but Clyde is not

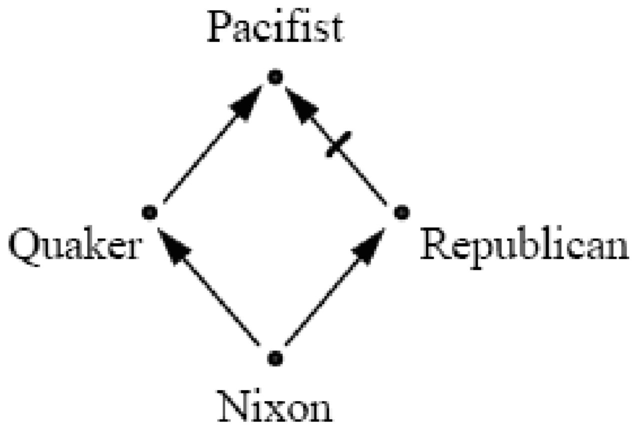
- Inherited properties do not always hold, and can be overridden (defeated)

- Conclusions determined by searching upward from **focus node** and selecting first version of property you want

Shortest Path Heuristic

- Links have polarity (positive or negative)
- Use **shortest path heuristic** to determine which polarity counts
- As a result, not all paths count in generating conclusions
- Some are ”preempted” but some are ”admissible”
- Think of paths as arguments in support of conclusions

Problem: Ambiguity

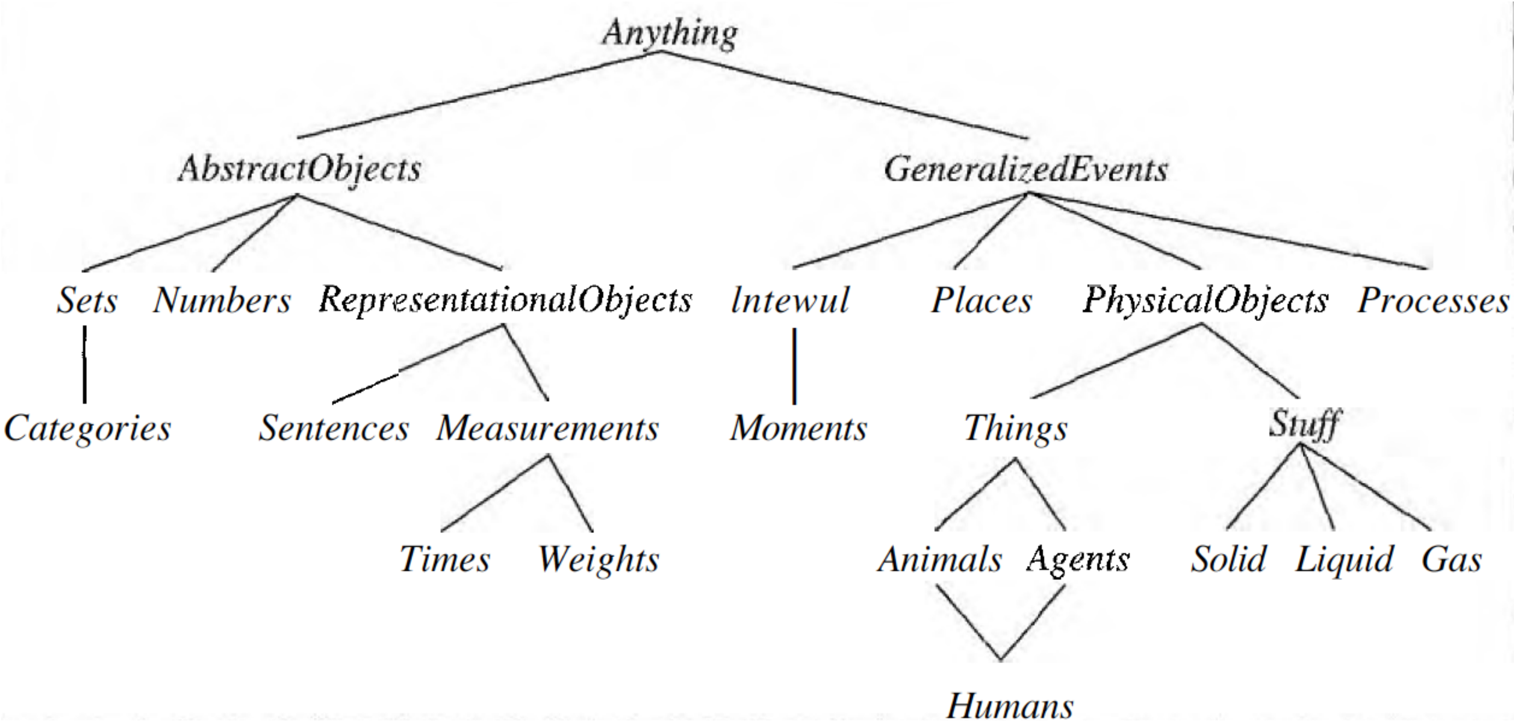


- There may be no single shortest path

⇒ Explicit handling of ambiguous reasoning chains

  - distinguish between ambiguous and unambiguous chains
  - preference for some extensions over others (default logic)
  - credulous vs. skeptical reasoning

Ontologies



- Organize knowledge about everything in a single taxonomy

frames

Simple Relational Knowledge

- We often want represent a large number of facts that follow a simple pattern

Planet	Star system	Radius	Moons
Mercury	Sun	2440 km	0
Venus	Sun	6052 km	0
Earth	Sun	6371 km	1
Mars	Sun	3389 km	2
Kepler-438b	Kepler-438	7135 km	?

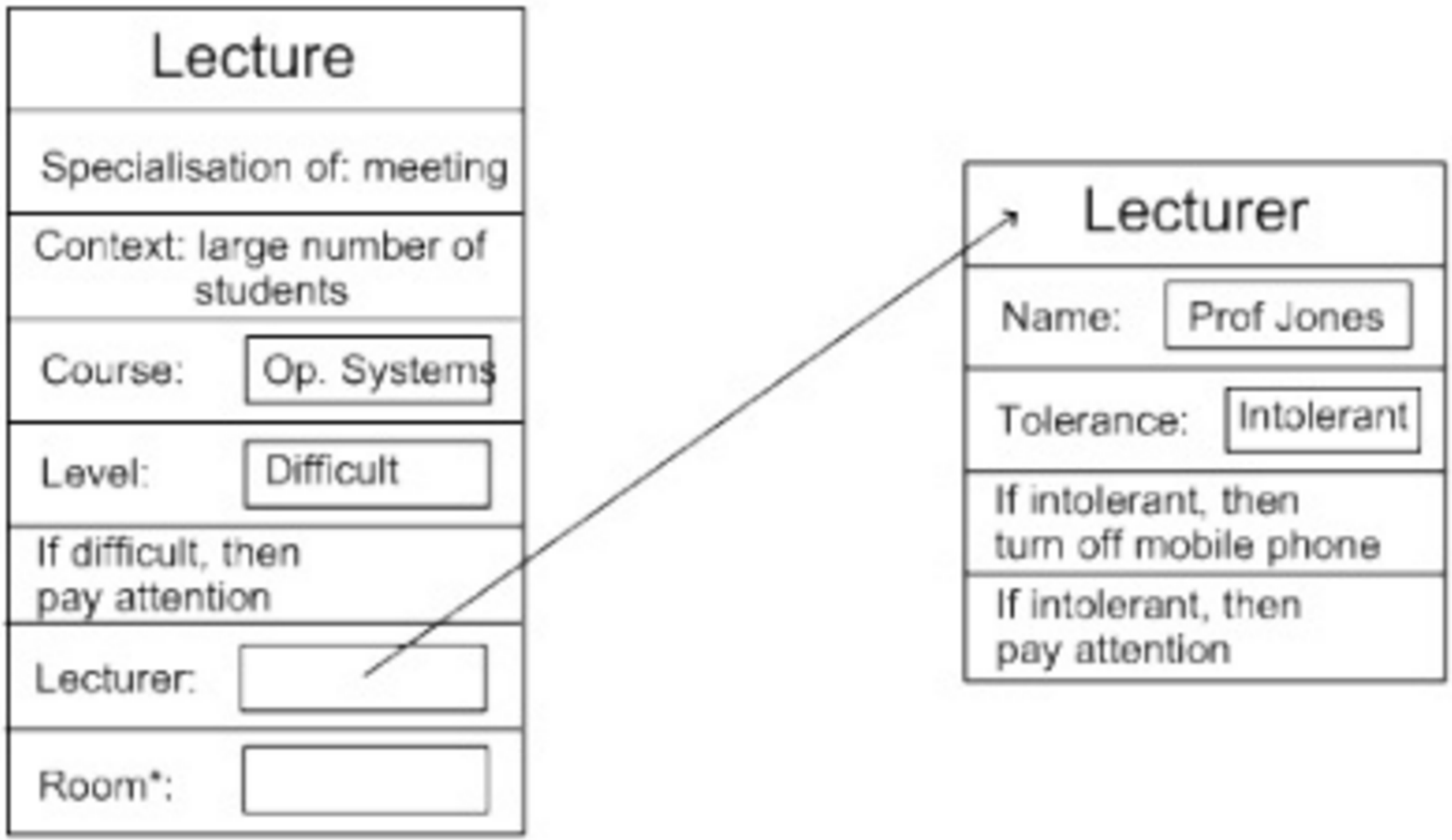
- Database table in relational database

Frames

- A frame is a collection of attributes or slots and associated values that describe some real world entity

- Each frame represents
  - a class, or
  - an instance (an element of a class)

Frames: Example



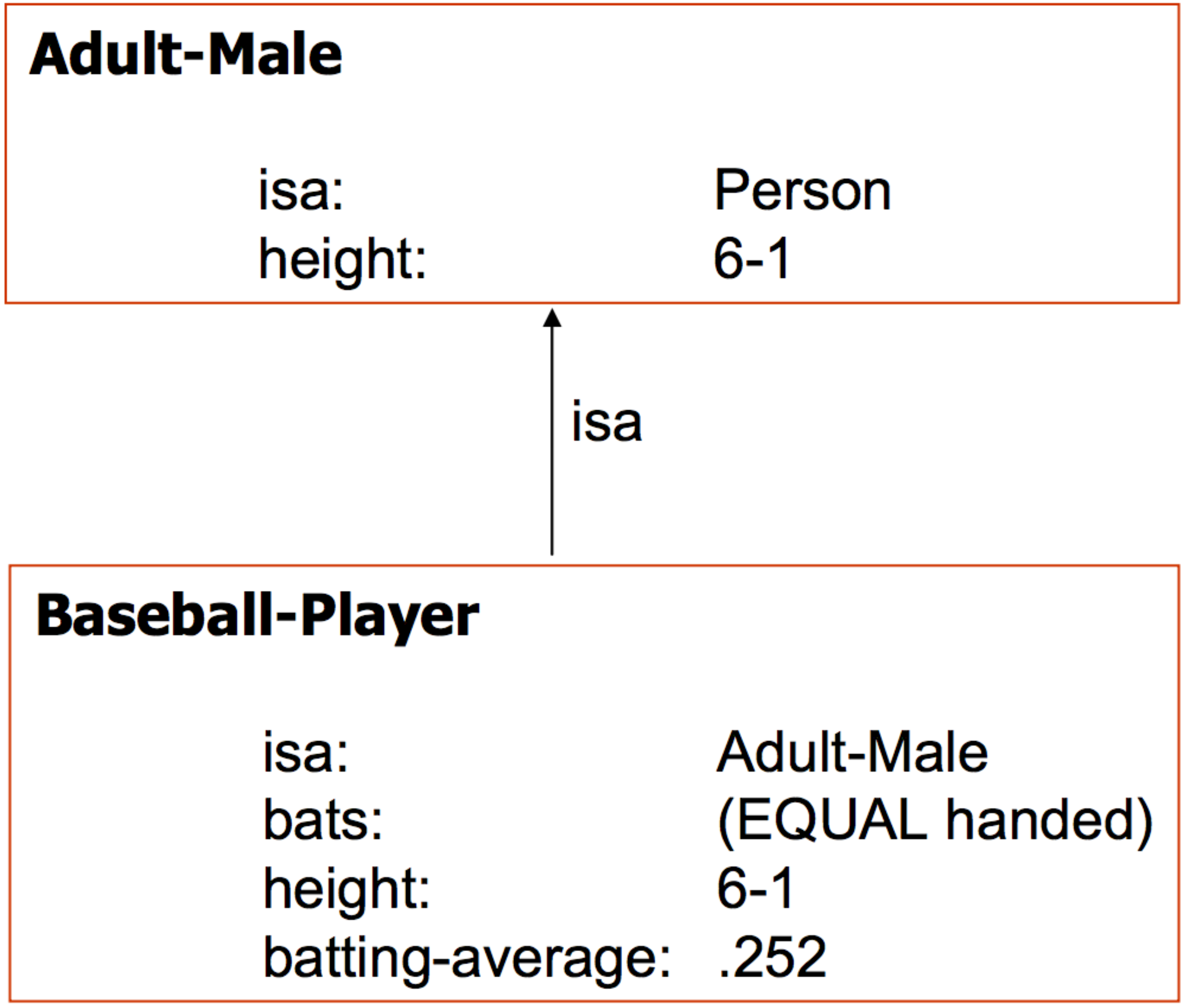
Knowledge Discovery

- Information retrieval when facing a new situation
  - information is stored in frames with slots
  - some of the slots trigger actions, causing new situations
- Frames are **templates**
  - need to be filled-in in a situation
  - filling them causes the agent to undertake actions and retrieve other frames
- Frames are extensions of record datatype in databases
- Also very similar to object oriented processing

Flexibility in Frames

- Slots in a frame can contain
  - information for choosing a frame in a situation
  - relationship between this and other frames
  - procedures to carry out after various slots filled
  - default information to use when input is missing
  - blank slots — left blank unless required for a task
  - other frames, which gives a hierarchy

Example: Frames Hierarchy



events

Events

- So far, facts were treated as true independent of time
- Events: need to describe what is true, when something is happening
- For instance: Flying event
  - $E \in Flyings$
  - $Flyer(E, Shankar)$
  - $Origin(E, SanFrancisco)$
  - $Destination(E, Baltimore)$
- The event may or may not ongoing during a specific time t: $Happens(E, t)$
- In general, facts that are true only at specific time points are called **fluents** e.g., $At(Shankar, Baltimore)$

Predicates of Events

- $T(f,t)$ — Fluent f is true at time t
- $Happens(e,i)$ — Event e happens over the time interval i
- $Initiates(e,f,t)$ — Event e causes fluent f to start at time t
- $Terminates(e,f,t)$ — Event e causes fluent f to end at time t
- $Clipped(e,f,i)$ — Fluent f ceases to be true at some point during time interval i
- $Restored(e,f,i)$ — Fluent f becomes true at some point during time interval i

Time Intervals

- There are a lot benefits to represent time in terms of intervals
  - moments: zero duration
  - extended intervals: positive time duration

- Allows the definition of
  - time interval meeting $End(i_1) = Start(I_2)$
  - time interval preceding another
  - during: time interval subset of other
  - overlap: time interval intersect, but neither is strict subset
  - beginning, end, indentiy of time intervals

- Example: $President(USA, t)$ matches different persons for different t

Scripts

- Definition

A script is a structured representation describing
a stereotyped sequence of events in a particular context.

- Scripts are used to organize events in knowledge bases

- Scripts are very related to the idea of frames

Components of a Script

- A script is composed of several components

- **Entry conditions** that must be true for the script to be called
- **Results or facts** that are true once the script has terminated
- **Props or the "things"** that make up the content of the script
- **Roles** are the actions that the individual participants perform
- **Scenes** which present temporal aspects of the script

Canonical Example: Restaurant Visit

- **Objects:** tables, menu, food, check, money, ...
- **Roles:** customer, waiter, cook, cashier, owner, ...
- **Entry conditions:** customer hungry, customer has money
- **Results:** customer not hungry, customer has less money, owner more money, ...
- **Scenes:**
  - Scene 1: Entering
    - customer enters restaurant
    - customers looks at tables
    - customer decides where to sit
    - ...
  - Scene 2: Ordering
    - waiter brings menu
    - ...
  - ...

Script Actions

Describing a script a special symbols of actions are used:

Symbol	Meaning	Example
ATRANS	transfer a relationship	*give*
PTRANS	transfer physical location of an object	*go*
PROPEL	apply physical force to an object	*push*
MOVE	move body part by owner	*kick*
GRASP	grab an object by an actor	*grasp*
INGEST	ingest an object by an animal	*eat*
EXPEL	expel from an animal's body	*cry*
MTRANS	transfer mental information	*tell*
MBUILD	mentally make new information	*decide*
CONC	conceptualize or think about an idea	*think*
SPEAK	produce sound	*say*
ATTEND	focus sense organ	*listen*

Detailed Script

Script Restaurant

Props
- Tables
- Menu
- F = Food
- Check
- Money

Roles
- P = Customer
- O = Waiter
- V = Cook
- K = Cashier
- S = Owner

Entry conditions
- P is hungry
- P has money

Results
- P has less money
- P is not hungry
- P is pleased (optional)
- S has more money

Scene 1: Entering
P PTRANS P into restaurant
P ATTEND eyes to tables
P MBUILD where to sit
P PTRANS P to table
P MOVE P to sitting position

Scene 2: Ordering
(Menu on table)
O brings menu)
P PTRANS menu to P

(S asks for menu)
S MTRANS signal to O
O PTRANS O to table
P MTRANS "need menu" to O
O PTRANS O to menu

O PTRANS O to table
O ATRANS menu to P

P MTRANS food list to P
* P MBUILD choice of F
P MTRANS signal to O
O PTRANS O to table
P MTRANS 'I want F' to O

O PTRANS O to V
O MTRANS (ATRANS F) to V

V MTRANS 'no F' to O
O PTRANS O to P
O MTRANS 'no F' to P
(go back to *) or
(go to Scene 4 at no pay path)

V DO (prepare F script)
to Scene 3

Scene 3: Eating
V ATRANS F to O
O ATRANS F to P
P INGEST F

Option: Return to Scene 2 to order more; otherwise, go to Scene 4

Scene 4: Exiting
P MTRANS to O
(O ATRANS check to P)

O MOVE write check
O PTRANS O to P
O ATRANS check to P
P ATRANS tip to O
P PTRANS P to K
P ATRANS money to K
P PTRANS P to out of restaurant

No pay path

Schank un Abelson, 1977

cyc

Cyc

- Goal: codify millions of pieces of knowledge that compose common sense
- Name "Cyc" from "encyclopedia"
- History
  - 1984: started by Microelectronics and Computer Technology Corporation
  - 1986: estimated effort to complete Cyc 250,000 rules and 350 man-years
  - 1994: spun off into Cycorp, Inc.
  - 2008: links to Wikipedia articles
  - 2012: publicly available OpenCyc
- Basic structure
  - facts such as "Every tree is a plant" and "Plants die eventually"
  - inference to deduce "Trees die eventually"
  - CycL language: predicate calculus (similar to that of the Lisp)
- Currently efforts to connect Cyc to natural language

Basics

- Collections
- Individual objects
- Relationships, e.g.
  - `#$isa` = instance of
  - `#$genls` = subclass of
- Operators
  - basic Boolean: `#$and`, `#$or`, `#$not`, `#$implies`, ...
  - quantifies: `#$thereExists`
  - etc.

Cyc Ontology

- Upper level
  - contains most broad abstract concepts, universal truths
  - smallest, but most widely referenced area of Cyc
- Middle level
  - not universal, but widely used abstraction layer
  - e.g., geospatial relationships, broad knowledge of human interaction
- Lower level
  - specific knowledge
  - e.g., information about chemistry, biology

Upper Level

- Encoded knowledge, e.g.
  - `(isa Event Collection)`
  - `(genls Event Situation)` (generalized = subset)
  - `(disjointWith Event PositiveDimensionalThing)`
  - `(genls HelicopterLanding Event)`

- Inferred knowledge
  - `(genls (BecomingFn Intoxicated) Event)`
  - `(relationExisistAll victim Event Victiom-UnfortunatePerson)`

Middle Level

- For instance, facts about human interaction
  - `(disjointWith SocialGathering SingleDoerAction)`
  - `(disjointWith SocialGathering ConflictEvent)`

- Properties of events
  - `(requiredActorSlots SocialGathering attendees)`

Lower Level

- For instance, chemistry
  - `(kegenlsStrongSuggestionPreds-RelationAllExists ChemicalReaction catalyst)`
  - `(genls ChemicalReaction PhysicalTransformationEvent)`
  - `(genls CombustionReaction ChemicalReaction)`
  - `(genls ExothermicReaction ChemicalReaction)`
  - `(genls ChemicalBonding ChemicalReaction)`
  - `(outputsCreated-TypeType CombustionReaction Flame)`

Example

- Want to encode very specific knowledge
  - `(eventOcurrsAt BruningOfPapalBull CityofWittenburgGermany)`
  -
    ```
    (dateOfEvent BruningOfPapalBull
        (DayFn 10
           (MonthFn December
              (YearFn 1520))))
    ```
  - `(attendee BruningOfPapalBull MarthinLuther-ReligiousFigure)`
  -
    ```
    (relationInstanceExistsMin BruningOfPapalBull
           attendees UniversityStudent 40)
    ```

- Can draw of fact that `MarthinLuther-ReligiousFigure` is already in Cyc

⇒ Various facts are connected (birth and death dates, country of residence, etc.)

expert systems

Expert Systems

- Build specialized system based on expert knowledge

- Recipe
  - interview expert about domain knowledge
  - formalize knowledge with logic rules

  ⇒ system that can replicate expert reasoning

- Advanced methods to incorporate uncertainty, etc.

Advantages

- Writing rules simpler than programming, may be done by experts themselves
- Rules can be easily inspected and modified
- Since system is based on logic rules, it provides an explanation

Disadvantages

- Experts may not understand their own intuitive reasoning process
- Some knowledge is hard to encode in logic rules
- Experts may not be motivated to help to get replaced by a system

semantic web

Distributed Knowledge

- Knowledge about the world is distributed

- World wide web
  - information from wide range of providers
  - target consumers: humans
  - format: pages in HTML
  - integration and reuse very limited

⇒ Need for ”machine-readable” web

A Smarter Web

- Find data sets from different places
- Take and aggregate data
- Analyze data in straightforward way

- Do all this automatically

Example

- I am a researcher

- I published a lot of papers
  - title, year, publication, presentation venue, page count, abstract, keywords, ...

  → need to make this information widely available

- Old solution: find someone who maintains a central repository

- Semantic web solution: define properties in XML schema on my web site

  → need properly defined XML schema

RDF: Resource Description Framework

- XML Markup lamguage that describes what is on the web

```
<rdf:Description rdf:about="http://bu.ch/123.html ">
  <author>
     <rdf:Description>
       <surname>Doe</surname>
       <firstname>John</firstname>
     </rdf:Description>
  </author>
  <title>My Life</title>
</rdf:Description>
```

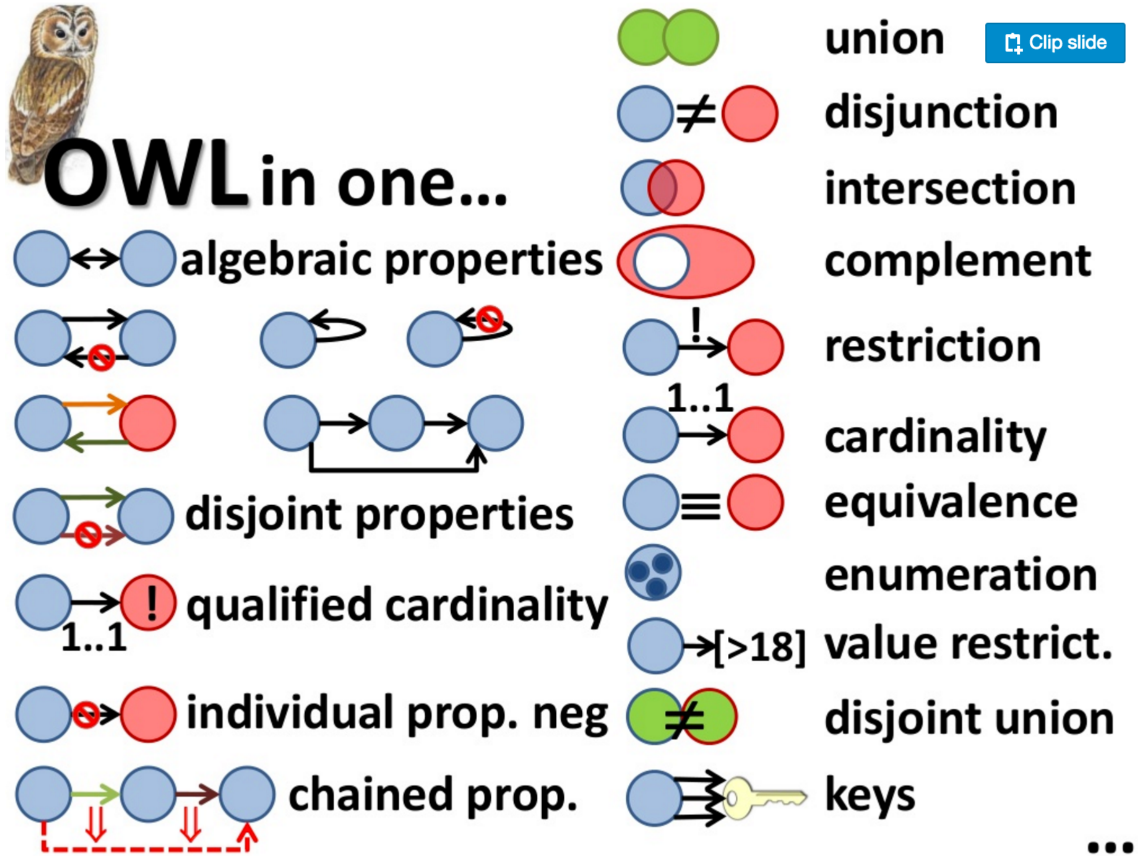
- Different schemas evolve
  → one wins out or mapping functions are defined

Querying Linked Open Data

- Various individuals and organizations make data available
- SPARQL: query protocol to access this data
  - query language
  - result format
  - access protocol
- Example: persons at least 18-year old

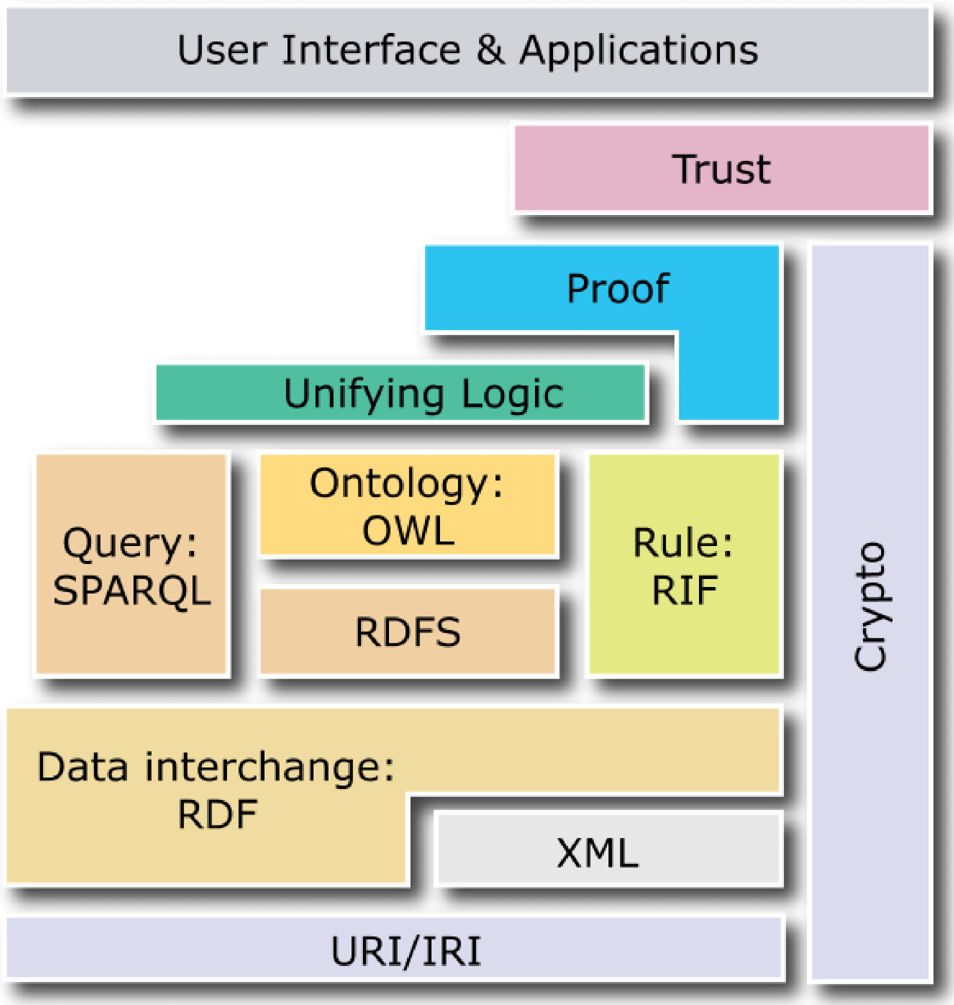
```
PREFIX ex: <http://inria.fr/schema#>
SELECT ?person ?name
WHERE {
  ?person rdf:type ex:Person .
  ?person ex:name ?name .
  ?person ex:age ?age .
  FILTER (?age > 17)
}
```

Ontologies



- Schemas need to be connected in shared ontology
- OWL: provides primitives for complex ontologies

Layers of the Semantic Web



Summary

- Basic principles of knowledge:
  objects, categories, events, beliefs, ...
- Need for formal knowledge representation systems
  - inheritance and semantic networks
  - frames and scripts
- Practical efforts to encode knowledge
  - Cyc: 40 year centralized effort
  - semantic web: open linked data with public protocols