INTRODUCTION TO KNOWLEDGE REPRESENTATION

Basi di Dati e Basi di Conoscenza

Roberto Basili

(many slides from Paula Matuszek Spring, 2010)

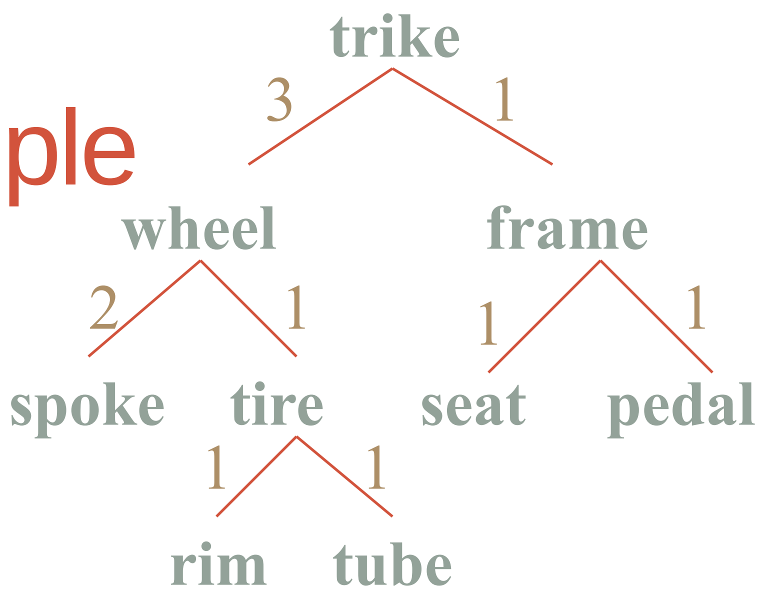
Motivation: what DB cannot do

- SQL-92 cannot express some queries:
  - Are we running low on any parts needed to build a ZX600 sports car?
  - What is the total component and assembly cost to build a ZX600 at today's part prices?
- Can we extend the query language to cover such queries?
  - Yes, by adding recursion.

A different perspective on querying DBs: Datalog

- SQL queries can be read as follows:
  "**<u>If</u>** some tuples exist in the From tables that satisfy the Where conditions,
  <u>then</u> the Select tuple is in the answer."
- Datalog is a query language that has the same **<u>if-then</u>** flavor:
  - **New**: The answer table can appear in the From clause, i.e., be defined recursively.
  - *Prolog* style syntax is commonly used.

Example



- Find the components of a trike?
- We can write a relational algebra query to compute the answer on ***the given instance of Assembly***.
- But there is no R.A. (or SQL-92) query that computes the answer on ***all Assembly instances***.

part	subpart	number
trike	wheel	3
trike	frame	1
frame	seat	1
frame	pedal	1
wheel	spoke	2
wheel	tire	1
tire	rim	1
tire	tube	1

Assembly instance

The Problem with R.A. and SQL-92

- Intuitively, we must join Assembly with itself to deduce that trike contains spoke and tire.
  - Takes us one level down Assembly hierarchy.
  - To find components that are one level deeper (e.g., rim), need another join.
  - To find all components, need as many joins as there are levels in the given instance!
- For any relational algebra expression, we can create an Assembly instance for which some answers are not computed by including more levels than the number of joins in the expression!

Introduction

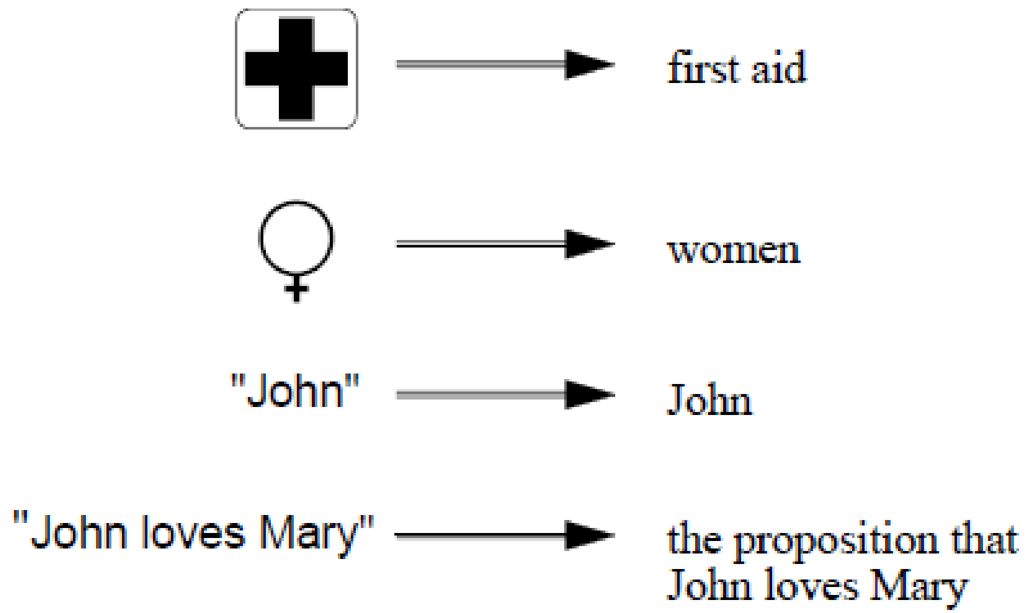
- Knowledge Representation means:
  - Capturing human knowledge
  - In a form computer can reason about
- Why?
  - Model human cognition
  - Add power to search-based methods
- Actually a component of all software development

KR Introduction

- General problem in Computer Science
- Solutions = Data Structures
  - words, arrays
  - records
  - lists, queues
  - objects
- More specific problem in AI
- Solutions = knowledge structures
  - decision trees
  - logic and predicate calculus
  - rules: production systems
  - description logics, semantic nets, frames
  - scripts
  - ontologies

The notion of Representation

Symbols standing for things in the world



Knowledge representation:

symbolic encoding of propositions believed
(by some agent)

Reasoning

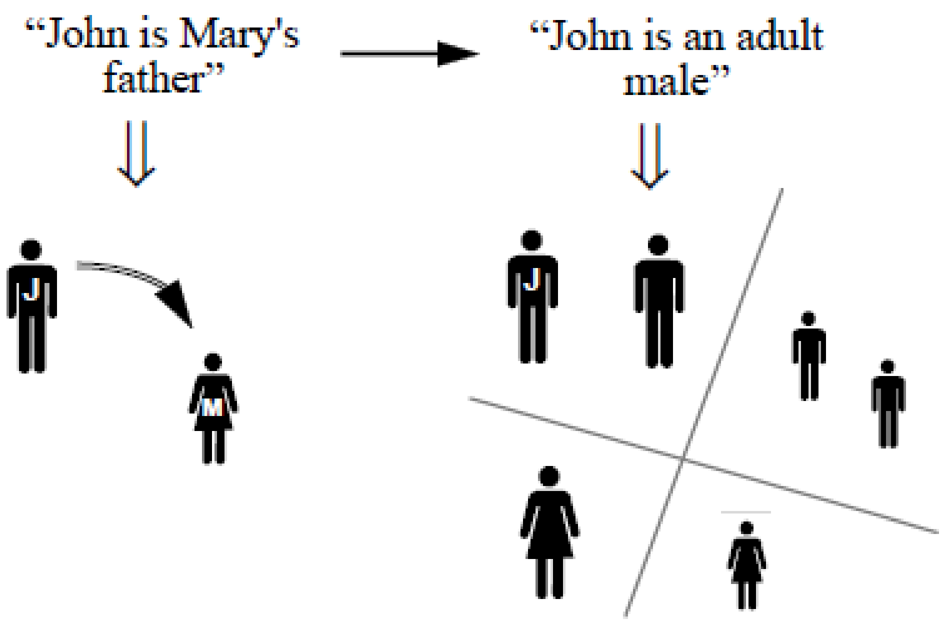
Manipulation of symbols encoding propositions to produce representations of new propositions

Analogy: arithmetic

"1011" + "10" → "1101"

⇓ ⇓ ⇓

eleven two thirteen



Forms of Logic Representation

```
printColor(snow) :- !, write("It’s white.").
printColor(grass) :- !, write("It’s green.").
printColor(sky) :- !, write("It’s yellow.").
printColor(X) :- write("Beats me.").
```

Here is an alternate:

```
printColor(X) :- color(X,Y), !,
       write("It’s "), write(Y), write(".").
printColor(X) :- write("Beats me.").
color(snow,white).
color(sky,yellow).
color(X,Y) :- madeof(X,Z), color(Z,Y).
madeof(grass,vegetation).
color(vegetation,green).
```

Forms of Logic representation

- Why is the second Prolog program a more adequate knowledge representation of the underlying procedure than the first one?
  - Epistemologically transparent
  - Clear distinction on *what is known* and *what is to be done*
  - Does not depend on what the system *believes* about printing but rather on *what the system knows about snow and colours*

Forms of Logic representation

- Only the second program has explicit representation of ‘knowledge’ that snow is white
- The second program does what it does when asked for the colour of *snow* because of this knowledge. When `colour(snow,white)` is removed, it will not print the right colour for snow.
- What makes the system knowledge-based is NOT
  - the use of a particular logical-looking language like Prolog
  - or having representation of true facts (`colour(sky, yellow)` is false)
  - or having lots of facts, or having a complex structure
- rather, *it is having explicit representation of knowledge which is used in the operation of the program*

Advantages of Logic representation

- The second Prolog program IS a more adequate knowledge representation of the underlying procedure
- **Advantages**
  - **Modularity** helps in REUSING the knowledge for other tasks
  - **Refining strategies** in printing out from colours (e.g. new beliefs)
  - **Cheaper mantainance** at software level (if something is wrong, e.g. grass is not green a minimal set of facts must be changed)
  - The system seems able to **explain** what it does

Knowledge-based systems

- Knowledge-based systems are systems for which intentional stance is grounded by design in symbolic representation
- The symbolic representation of knowledge is called a knowledge base.

- One possible implementation:
  - **Representation**: as a set of sentences of first order logic
  - **Reasoning**: deducing logical consequences

Characteristics of a good KR:

- It should
  - Be able to represent the knowledge important to the problem
  - Reflect the structure of knowledge in the domain
    - Otherwise our development is a constant process of distorting things to make them fit.
  - Capture knowledge at the appropriate level of granularity
  - Support incremental, iterative development
- It should *not*
  - Be too difficult to reason about
  - Require that more knowledge be represented than is needed to solve the problem

Kinds of Knowledge

Things we need to talk about and reason about; what do we know?

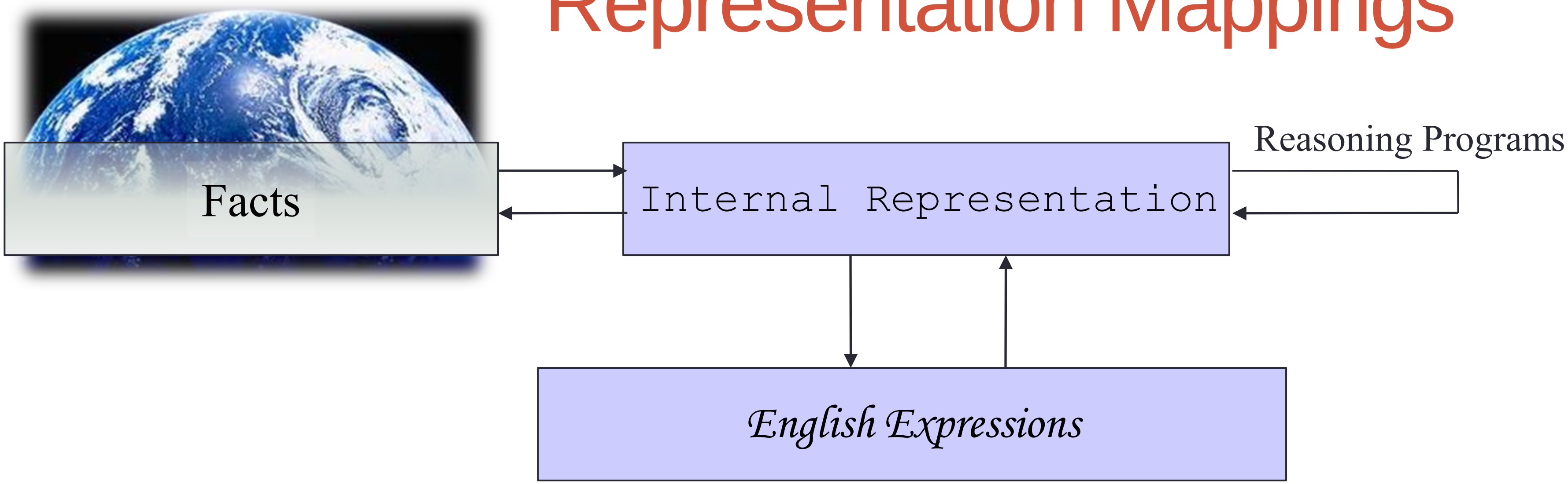
- Objects
  - Descriptions: what the object is
  - Classifications: how can we distinguish it from other objects
- Events
  - Time sequence
  - Cause and effect
- Relationships
  - Among objects
  - Between objects and events
- Meta-knowledge: Why? What cannot be said?

Distinguish between knowledge and its representation

Mappings are not one-to-one

Never get it complete or exactly right

Representation Mappings

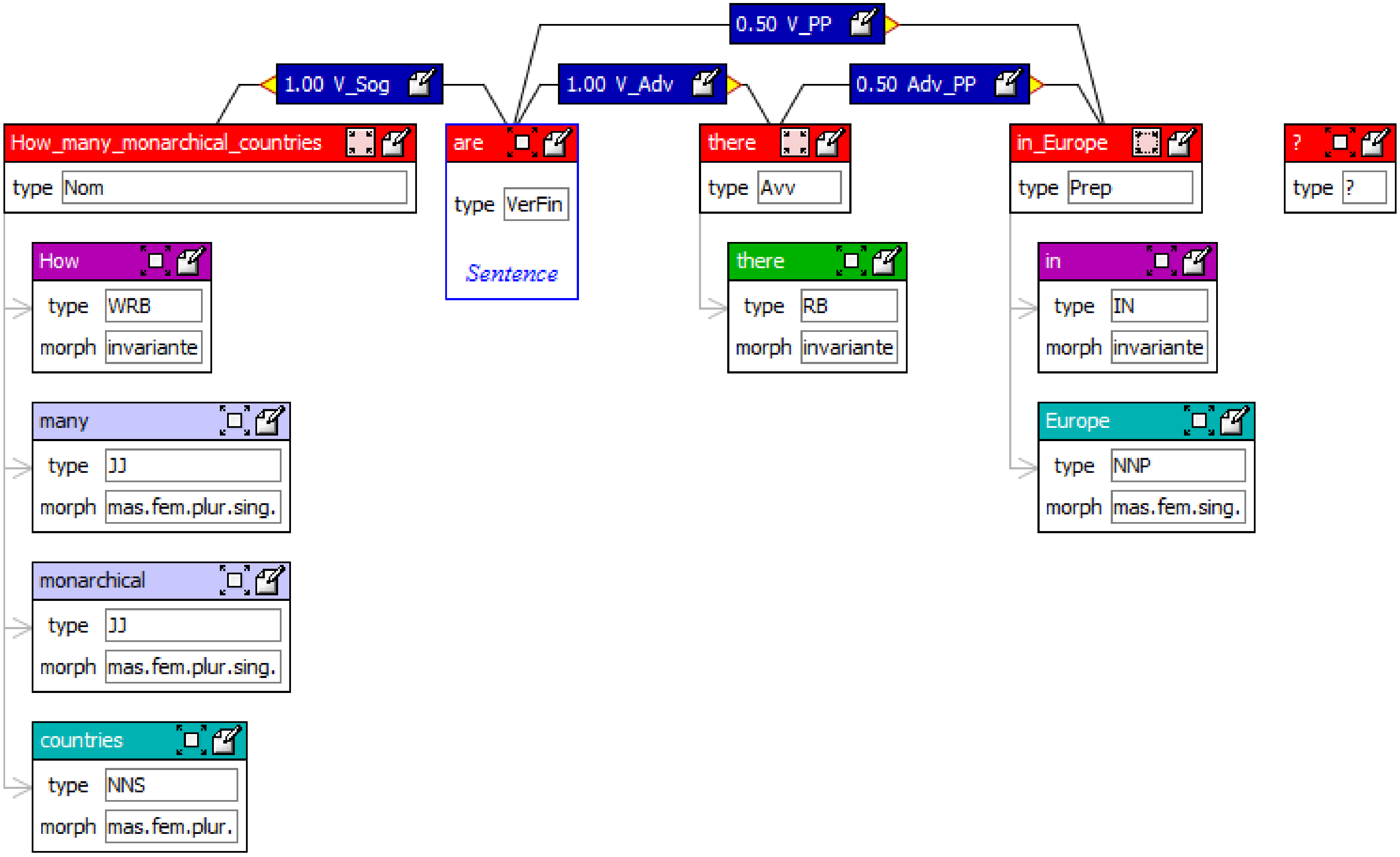


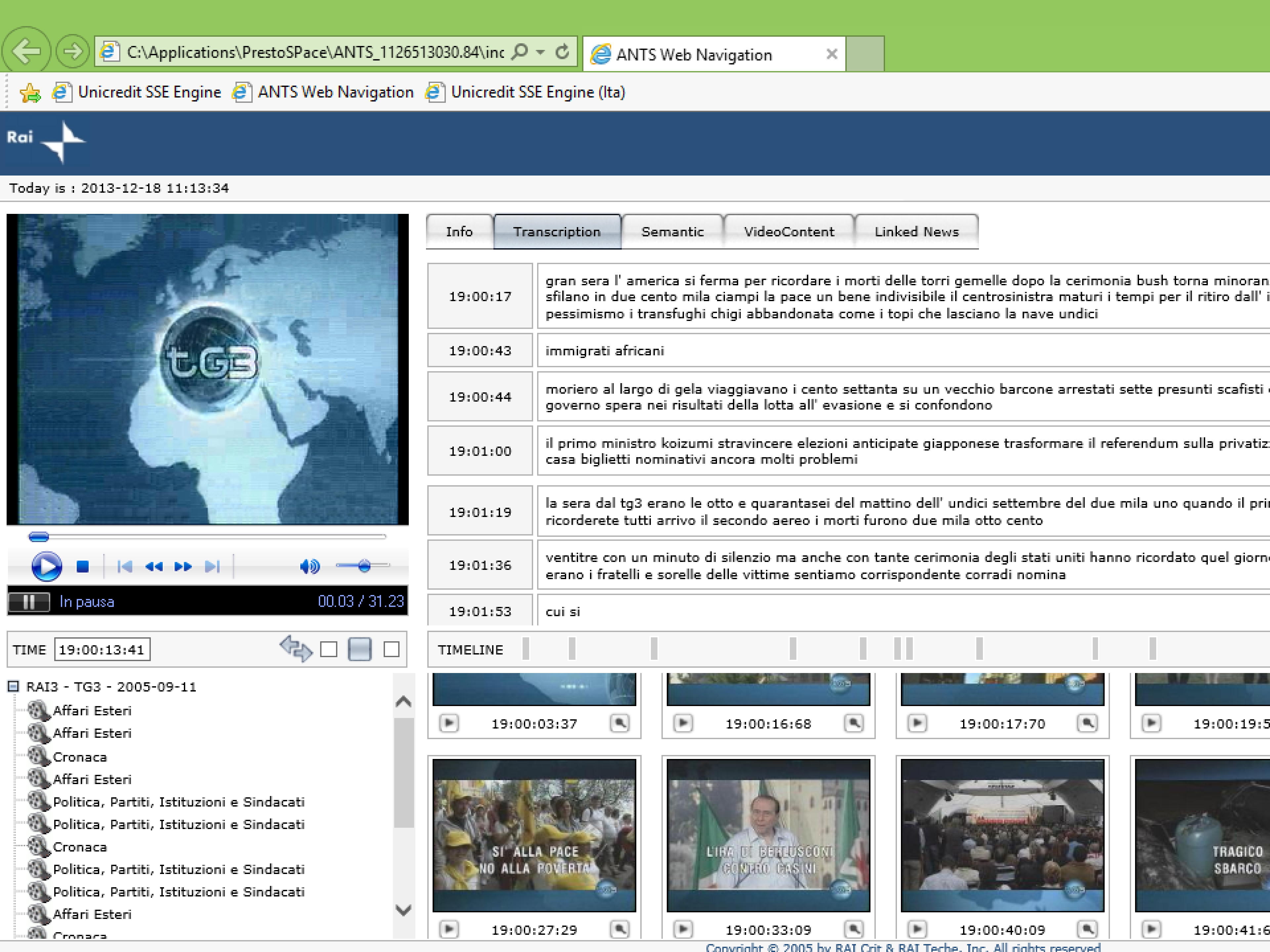
- Knowledge Level: entailment, consistency, complexity of reasoning
- Symbol Level: data structures, computational architecture
- Mappings are not one-to-one
- Never get it complete or exactly right

Knowledge engineering!

- Modeling the “right” conditions and the “right” effects at the “right” level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
    - Our intelligent systems should be able to learn about the conditions and effects, just like we do!
    - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Knowledge-based systems: NL parser




C:\Applications\PrestoSPace\ANTS_1126513030.84\inc
ANTS Web Navigation
Unicredit SSE Engine
ANTS Web Navigation
Unicredit SSE Engine (Ita)
Rai
Today is : 2013-12-18 11:13:34
Info
Transcription
Semantic
VideoContent
Linked News
19:00:17
gran sera l' america si ferma per ricordare i morti delle torri gemelle dopo la cerimonia bush torna minoran
sfilano in due cento mila ciampi la pace un bene indivisibile il centrosinistra maturi i tempi per il ritiro dall' i
pessimismo i transfughi chigi abbandonata come i topi che lasciano la nave undici
19:00:43
immigrati africani
19:00:44
moriero al largo di gela viaggiavano i cento settanta su un vecchio barcone arrestati sette presunti scafisti
governo spera nei risultati della lotta all' evasione e si confondono
19:01:00
il primo ministro koizumi stravincere elezioni anticipate giapponese trasformare il referendum sulla privatiz
casa biglietti nominativi ancora molti problemi
19:01:19
la sera dal tg3 erano le otto e quarantasei del mattino dell' undici settembre del due mila uno quando il pri
ricorderete tutti arrivo il secondo aereo i morti furono due mila otto cento
19:01:36
ventitre con un minuto di silenzio ma anche con tante cerimonia degli stati uniti hanno ricordato quel giorn
erano i fratelli e sorelle delle vittime sentiamo corrispondente corradi nomina
19:01:53
cui si
tG3
In pausa
00.03 / 31.23
TIME
19:00:13:41
TIMELINE
RAI3 - TG3 - 2005-09-11
Affari Esteri
Affari Esteri
Cronaca
Affari Esteri
Politica, Partiti, Istituzioni e Sindacati
Politica, Partiti, Istituzioni e Sindacati
Cronaca
Politica, Partiti, Istituzioni e Sindacati
Politica, Partiti, Istituzioni e Sindacati
Affari Esteri
Cronaca
19:00:03:37
19:00:16:68
19:00:17:70
19:00:19:5
SI' ALLA PACE
NO ALLA POVERTA'
L'IRA DI BERLUSCONI
CONTRO CASINI
TRAGICO
SBARCO
19:00:27:29
19:00:33:09
19:00:40:09
19:00:41:6
Copyright © 2005 by RAI Crit & RAI Teche, Inc. All rights reserved

Some Typical Paradigms of KR

- Logic and predicate calculus
  - Knowledge (e.g. facts) as logical formulas and inference as deduction
- Rules: production systems
  - Facts as always true rules (only LHS)
  - Rules as RHS <= LHS pairs
- Semantic nets and frames
  - Networks of concepts, arcs as inferences
  - Description logics
- Scripts: procedural extensions
- Ontologies: terminological dictionaries plus inference

Logic and Predicate Calculus

- Very rich representation
- For big real-world problems has some significant issues:
    - very bushy inference
    - does not match human expert thinking very well
        - excluded middle
        - No good choice for “don’t know”

```
/* Facts */
color(snow,white). %the snow is white
color(sky,yellow). %the sky is yellow

/* Inference Rules */
printColor(X) :-
 color(X,Y), !,    % as Y is the color of X
 write('It\'s '), % … print this
 write(Y),
 write('.').       %to proof the thesis
```

Logic Representation: an example

```
/* Facts */
color(snow,white).          %the snow is white
color(sky,yellow).          %the sky is yellow

/* Inference Rules */
color(X,Y) :-               % Y  is the color of X
 madeof(X,Z),               % IF  X I  s made of Z
 color(Z,Y).                % AND  Y is the color of Z
```

Logic Representation: an example

Production Rules

- CLIPS, for instance
- Knowledge is represented as if-then rules:
  - if <condition> (LHS, left hand side)
  - then <action> (RHS, right hand side)
- If car won't start,
  then see if battery is dead.
- If a person is a student
  then a person has an ID card

Rule-based Inference: evaluation

- Advantages
  - Relatively fast
  - Captures natural human patterns
  - Modular
  - Can capture uncertainty and non-monotonicity
  - Restricted syntax simplifies editors, learning, etc.
- Disadvantages
  - Neither sound nor complete
  - Requires conflict resolution
  - restricted syntax reduces expressiveness
  - System behaviour reliant on conflict resolution strategy
  - adding new rules may produce unusual effects under conflict resolution

Structured Knowledge Representations

- Modeling-based representations reflect the structure of the domain, and then reason based on the model.
  - Semantic Nets
  - Frames
  - Scripts
- Sometimes called associative networks

Basics of Associative Networks

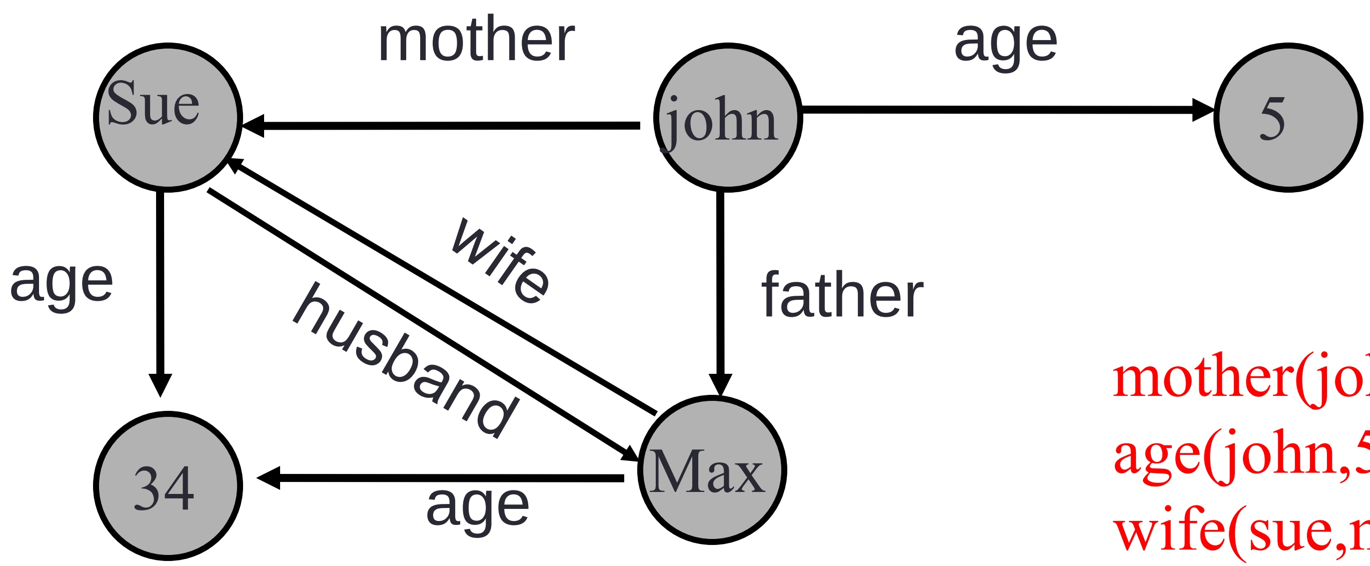
- All include
  - Concepts
  - Various kinds of links between concepts
    - “has-part” or aggregation
    - “is-a” or specialization
    - More specialized depending on domain
- Typically also include
  - Inheritance
  - Some kind of procedural attachment

Semantic Nets

- graphical representation for propositional information
- originally developed by M. R. Quillian as a model for human memory
- labeled, directed graph
- nodes represent objects, concepts, or situations
  - labels indicate the name
  - nodes can be instances (individual objects) or classes (generic nodes)
- links represent relationships
  - the relationships contain the structural information of the knowledge to be represented
  - the label indicates the type of the relationship

Nodes and Arcs

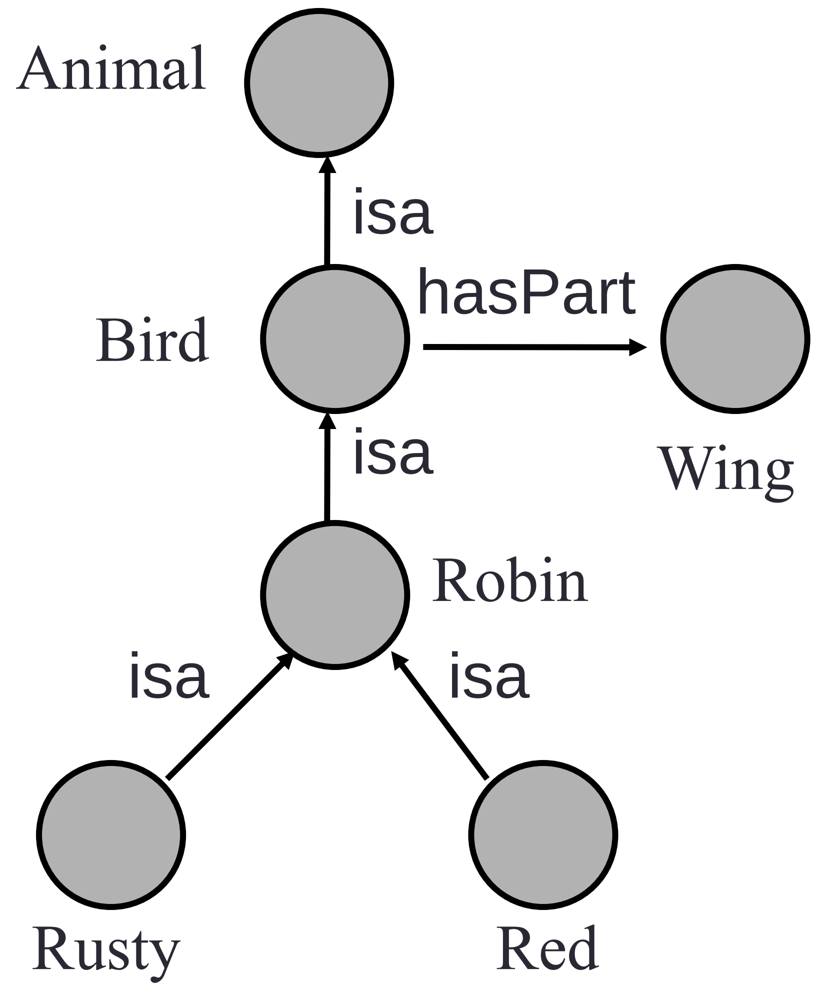
- Arcs define binary relationships that hold between objects denoted by the nodes.



mother(john,sue)
age(john,5)
wife(sue,max)
age(max,34)
…

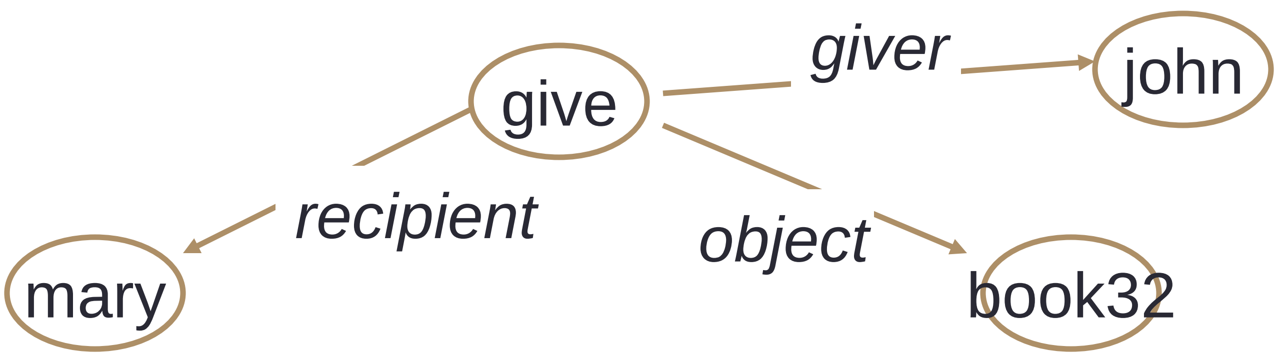
Semantic Networks

- The ISA (is-a) or AKO (a-kind-of) relation is often used to link instances to classes, classes to superclasses
- Some links (e.g. hasPart) are inherited along ISA paths.
- The semantics of a semantic net can be relatively informal or very formal
    - often defined at the implementation level



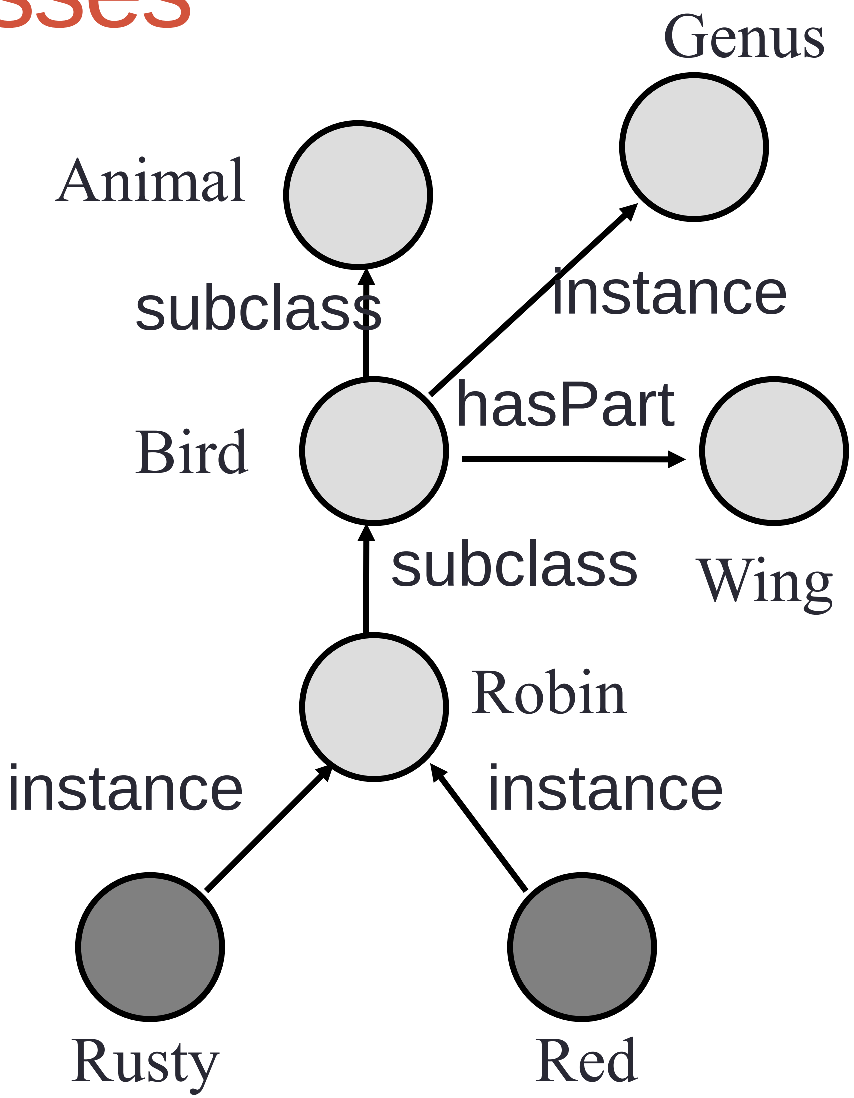
Reification

- Non-binary relationships can be represented by "turning the relationship into an object"
- This is an example of what logicians call "*reification*"
  - reify v : consider an abstract concept to be real
- We might want to represent the generic *give* event as a relation involving three things: a giver, a recipient and an object, **`give(john,mary,book32)`**



Individuals and Classes

- Many semantic networks distinguish
  - –nodes representing individuals and those representing classes
  - –the “subclass” relation from the “instance-of” relation



Inference by Inheritance

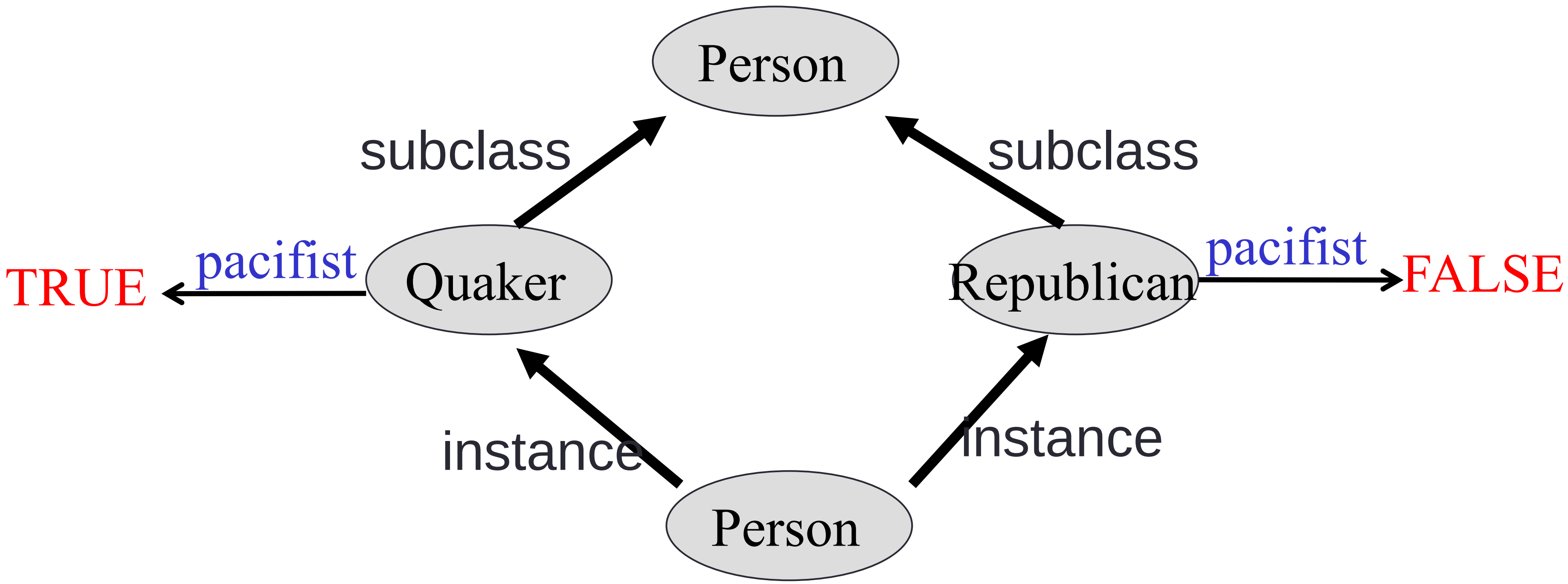
- One of the main kinds of reasoning done in a semantic net is the inheritance of values along the subclass and instance links.
- Semantic networks differ in how they handle the case of inheriting multiple different values.
  - All possible values are inherited, or
  - Only the "lowest" value or values are inherited

Multiple inheritance

- A node can have any number of superclasses that contain it, enabling a node to inherit properties from multiple “parent” nodes and their ancestors in the network.
- This can lead to conflicting inheritance.
- Some rules often used to determine inheritance in such “tangled” networks where multiple inheritance is allowed:
  - If $X<A<B$ and both A and B have property P, then X inherits A’s property.
  - If $X<A$ and $X<B$ but neither $A<B$ nor $B<Z$, and A and B have property P with different and inconsistent values, then X does not inherit property P at all.

Nixon Diamond

- This was the classic example circa 1980.

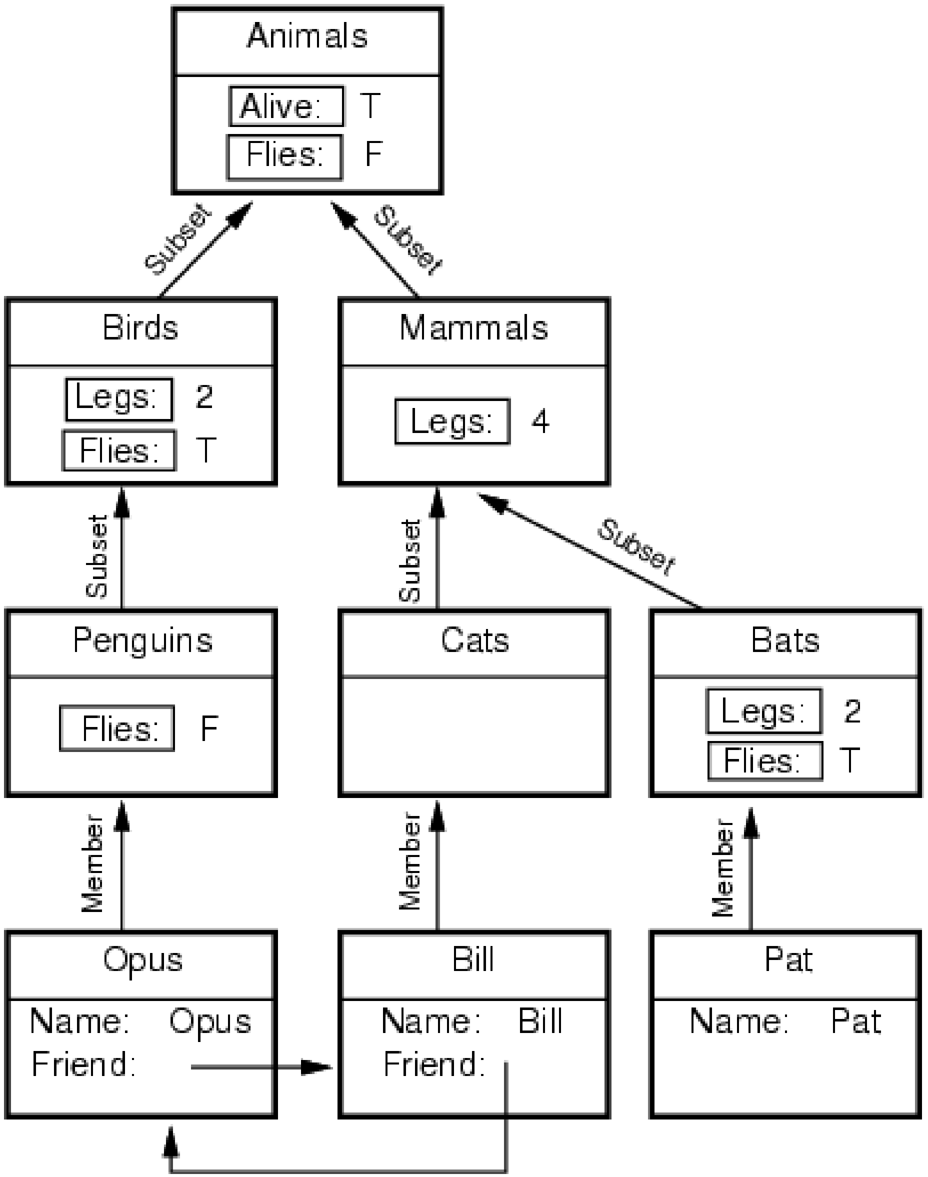


From Semantic Nets to Frames

- Semantic networks morphed into Frame Representation Languages in the ‘70s and ‘80s.
- A frame is a lot like the notion of an object in OOP, but has more meta-data.
- Represents related knowledge about a structured subject
- A frame has a set of slots.
- A slot represents a relation to another frame (or value).
- A slot has one or more facets.
- A facet represents some aspect of the relation.

Facets

- A slot in a frame holds more than a value.
- Other facets might include:
    - current fillers (e.g., values)
    - default fillers
    - minimum and maximum number of fillers
    - type restriction on fillers (usually expressed as another frame object)
    - attached procedures (if-needed, if-added, if-removed)
    - salience measure
    - attached constraints or axioms
- In some systems, the slots themselves are instances of frames.



Rel(Alive,Animals,T)
Rel(Flies,Animals,F)

Birds $\subset$ Animals
Mammals $\subset$ Animals

Rel(Flies,Birds,T)
Rel(Legs,Birds,2)
Rel(Legs,Mammals,4)

Penguins $\subset$ Birds
Cats $\subset$ Mammals
Bats $\subset$ Mammals
Rel(Flies,Penguins,F)
Rel(Legs,Bats,2)
Rel(Flies,Bats,T)

Opus $\in$ Penguins
Bill $\in$ Cats
Pat $\in$ Bats

Name(Opus,"Opus")
Name(Bill,"Bill")
Friend(Opus,Bill)
Friend(Bill,Opus)
Name(Pat,"Pat")

(a) A frame-based knowledge base

(b) Translation into first-order logic

Usage of Frames

- filling slots in frames
    - can inherit the value directly
    - can get a default value
    - these two are relatively inexpensive
    - can derive information through the attached procedures (or methods) that also take advantage of current context (slot-specific heuristics)
    - filling in slots also confirms that frame or script is appropriate for this particular situation

Example Frame: Low-level frame

Object	2o emicorso BdDC
ISA	Modulo
Date	Inverno, 2013
Time	9:00-11:30 Gio
Instructor	Rbasili
Exam	TRM
TA	De Cao

Frame Example: default frame

Object	CorsoUni
ISA	MODULO
Date	DATE
Time	TIME
Instructor	IF-NEEDED: Ask department IF-ADDED: Update payroll
Exam	TEST
TA	EXPERT

Description Logics

- Description logics provide a family of frame-like KR systems with a formal semantics.
  - E.g., KL-ONE, LOOM, Classic, …
- An additional kind of inference done by these systems is automatic classification
  - finding the right place in a hierarchy of objects for a new description
- Current systems take care to keep the languages simple, so that all inference can be done in polynomial time (in the number of objects)
  - ensuring tractability of inference

Description Logics

- Notations to make it easier to describe definitions and properties of categories
- Taxonomic structure is organizing principle
  - Subsumption: Determine if one category is a subset of another
  - Classification: Determine the category in which an object belongs
  - Consistency: Determine if membership criteria are logically satisfiable

Some DL Representations

- DAML+OIL
- RDF
- CYCL
- OWL

- Protege supports several of these;

Ontologies

- Structuring knowledge in a useful fashion
- An ontology formally represents concepts in a domain and relationships between those concepts
- The concept originated in philosophy; a model of a theory of nature or existence.
- An ontology describes the things we want to talk about, including both objects and relationships

Knowledge Engineering

- Process of representing domain knowledge formally
- Includes several components or phases:
  - Becoming familiar with the domain
  - Choosing a knowledge representation
  - Adding high-level knowledge
  - Adding more detailed knowledge
  - Testing Knowledge
  - Updating and maintaining the knowledge base
- The knowledge engineering bottleneck is a significant problem.

ENGINEERING FOR KR

R. Basili

KE: Familiarization -- process of becoming acquainted with domain for which the KB is being developed.

- General Domain
  - Typical goals or purpose
  - Training, skills needed
  - Typical sources of knowledge used
  - Typical information gathered
  - May include observations, interviews, training
- Specific Setting
  - Goal or purpose for this organization
  - Typical setup and roles
  - Materials and processes
  - Specific knowledge involved

Typical Familiarization Activities

- Interview sponsors of system: establish goals, scope, identify experts
- Interview experts: get general feel for their activities, clarify result expert system should produce
- Observe setting and experts
- Your goal at this stage is not to start capturing knowledge, it is to learn enough about the domain and the requirements to make some decisions about knowledge representation.

Knowledge Representation: choosing a representation appropriate for the domain

- Requires familiarity with the general domain
- Desirable characteristics
  - Easy to represent relevant knowledge
  - Doesn't require irrelevant knowledge
  - Easy to integrate new knowledge
  - Can reason about it appropriately and efficiently
  - Reflects the semantics of the domain
- Sometimes hard to assess initially; good to do a small proof of concept of what you choose

Domain Characteristics: Relevant to determining Knowledge Representation

- Kind of reasoning of human experts
- Level of complexity of domain
  - Inputs
  - Outcomes
  - Relationships between inputs and outcomes
- Kinds of knowledge
  - Structure
  - Heuristics
  - Inheritance
- Stability of knowledge

KE: High-Level Knowledge

- High level knowledge is
  - Knowledge about structure of domain
  - Organizing knowledge
  - Initial Information
- Tasks include
  - Preparation
  - Knowledge Acquisition
  - Knowledge Analysis
  - Coding
  - Testing
- Best carried out with sources who have an organized, coherent view of the entire domain, if available

KR: Detailed Knowledge

- More specific knowledge
- Lower level details
- Actual cases
- Expected outcomes for cases
- Steps
  - Preparation
  - Knowledge Acquisition
  - Knowledge Analysis
  - Coding
  - Testing

E: Detailed Knowledge -- Preparation

- Preparing for actual knowledge acquisition steps.
- The goal is to be ready to make optimal use of source's time
  - Review Materials
  - Become familiar with terminology
  - Collect sources
  - Training
  - Review any existing adjacent knowledge bases

KE: Detailed Knowledge -- Sources

- Sources can include
  - Textbooks and manuals
  - Reports
  - Databases, empirical data, case studies
  - Human experts
- Issues include
  - Scattered knowledge
  - Multiple sources of knowledge
  - Contradictory knowledge
  - Irrelevant knowledge
- A significant amount of the knowledge available is “noise” for the purposes of a specific system. The knowledge engineer needs skill at screening out the noise and organizing the remainder.

KE: Detailed Knowledge – Books and Manuals

- Usually easy to obtain
- Optimal use of documents typically includes
  - Aid to becoming familiar with terminology and general subject matter
  - Reference between sessions with an expert
  - Good source for diagrams, detailed names, etc
  - Source for online aids built in to expert system
- Documents used extensively by experts may be especially helpful
- Screen carefully – it's easy to get buried.

KE: Detailed Knowledge – DBs, Empirical Data, Case Studies

- May already exist
  - Failure analyses
  - Loan records
  - Part replacement records
  - Help-line logs
- Useful for
  - Identifying frequent problems
  - Identifying rare problems
  - Determining weights or salience for conflict resolution
  - Identifying relevant factors
  - Looking for missing pieces
- Typically not directly useful for knowledge engineering

KE: Eliciting Knowledge From Experts

- Activities:
  - Interviewing
  - On-site observations
  - Problem discussion
  - Problem analysis
- General Guidelines
  - The more competent the expert, often the less able to describe the expertise
  - Don't believe everything the expert says
  - Don't be your own expert
  - Know what kind of knowledge you're after
  - Remember what your system needs to do

Some Good Beginning Questions

- How do you do your job?
- Can you remember that last case you dealt with?
- What facts or hypotheses do you try to establish when thinking about a problem?
- What kind of things do you like to know about when you begin to think about a problem?
- What are you trying to accomplish when you begin a case?

Some Good Intermediate Questions

- What type of values can this object have?
- What range of values is permissible?
- Can you describe what you mean by that?
- Tell me more about...
- How is this achieved
- What do you do next?
- How does that relate to our topic?
- How/Why/When do you do that?

KE: On-Site Observations

- Unstructured: Observe the expert at work
- Advantages:
  - Real problems
  - Insight into complexity and typical patterns
  - “Insider’s” view
- Disadvantages
  - Not always feasible
  - Limited
  - Time consuming
- Good in early stages

KE: Problem Analysis

- Step through a series of problems and their solutions
- Use real problems
- At each step query expert for
  - Rationale
  - Hypotheses
  - Goals
- Probe for hows of each conclusion
- Good to help get started on detailed knowledge
- Also useful in late stages to check for factors which were missed

KE: Problem Discussion

- Pick a representative set of problems and discuss them informally. Focus on
  - Data needed to solve problems
  - Solutions which are acceptable (ie, “we’re done now”)
  - Subproblems which can be identified
  - Knowledge needed to solve problems
  - Explanations
- Provides a good deeper level of knowledge, helps expand concepts.
- Can provide too much information at the beginning of the knowledge engineering process.

Problem Discussion Questions

- Some sample questions
  - Could you talk me through that decision making process?
  - What questions did you ask?
  - What data did you gather?
  - In what order?
  - Why did you ask those questions?
  - What did that answer tell you?
  - How did you decide that you were finished with this case?

Encoding the Knowledge

- Capture the knowledge you have just acquired in the knowledge representation you have chosen
- As soon as possible after acquiring
- Plan on approximately 10 hours of coding for 1 hour of acquisition.
- If you have questions or get stuck, go on to another part and note problem for later.
- Reference materials are often helpful here.

Testing and Updating

- Testing Knowledge
  - Test yourself first
  - When you are satisfied, review system performance with expert
  - If outcome was incorrect, why?
  - Very useful to have a set of regression tests.
- Updating and maintaining the knowledge base
  - The domain will change. Plan for it.
    - Organize your KB carefully
    - Modularize your KB
    - Document your KB
  - Essential to have a set of regression tests

Knowledge Engineering

- Actually capturing the information from the human subject matter expert (SME) in any of these formats is difficult and time-consuming
  - An iterative process of add knowledge/test.
  - Often a knowledge engineer or ontological engineer works with the SME
  - “What is the system for?” is critical
- Automated learning of knowledge is a very active research field right now.