CSC 480: Artificial Intelligence

Dr. Franz J. Kurfess
Computer Science Department
Cal Poly

Course Overview

- Introduction
- Intelligent Agents
- Search
  - problem solving through search
  - informed search
- Games
  - games as search problems
- Knowledge and Reasoning
  - reasoning agents
  - propositional logic
  - predicate logic
  - **knowledge-based systems**
- Learning
  - learning from observation
  - neural networks
- Conclusions

Chapter Overview
Knowledge-Based Systems

- Motivation
- Objectives
- Evaluation Criteria
- Terminology
  - Data
  - Knowledge
  - Information
- Knowledge Engineering
  - Methodology
- Ontologies
- Example Domains
  - Electronic Circuits
- Important Concepts and Terms
- Chapter Summary

Motivation

- without a good knowledge representation scheme, reasoning becomes very difficult
- a balance must be found between expressiveness of the representation language, understandability, and efficiency of the inference mechanism
- it is good to have a general-purpose methodology that can be refined for specific domains or problems
  - for performance reasons, sometimes specific schemes, methods, or inference mechanisms are used

Objectives

- distinguish between different phases in the knowledge representation process
- learn to apply the concepts of predicate logic for the representation of knowledge
- understand the difficulties of finding a good knowledge representation scheme
  - general-purpose
  - domain- or task-specific
- distinguish the tasks and activities of a knowledge engineer from those of a system designer of programmer

Terminology

- Data
  - fixed relations between individual items
  - often arranged as vectors or arrays
  - interpretation is usually provided for the collection of data as a whole
- Knowledge
  - separate, possibly dynamic relations between individual items
  - interpretation must be done for individual items
- Information
  - generic term, used in a very general sense
  - precisely defined for information theory

Knowledge Engineering

- often performed by a *knowledge engineer*
  - intermediary between users, domain experts and programmers
  - must know enough about the domain
    - objects
    - relationships
  - must be comfortable in the representation language
    - encoding of objects and relationships
  - must understand the implementation of the inference mechanism
    - performance issues
    - explanation of results

Knowledge Engineering Methodology

- domain
  - objects, facts
- vocabulary
  - predicates, functions, constants in the language of logic
  - ideally results in an ontology
- background knowledge
  - general knowledge about the domain
  - specify the axioms about the terms in the ontology
- specific problem
  - description of the instances of concepts and objects that determine the problem to be investigated
- queries
  - requests answers from the knowledge base/inference mechanism

Example: Electronic Circuits

- domain
  - digital circuits, wires, gates, signals, input and output terminals
  - types of gates: *AND, OR, XOR, NOT*
- vocabulary
  - constants for naming gates: X_1, X_2, …
  - functions for gate types
    - *Type*(X_1) = *XOR*
  - functions for terminals
    - *Out*(1, X_1) for the only output of X_1
    - *In*(n, X_1) for the input n of X_1
  - predicates for connectivity
    - *Connected*(*Out*(1, X_1), *In*(2, X_2))
  - two objects and a function for the signal values
    - *On, Off*
    - *Signal*(*Out*(1, X_1))

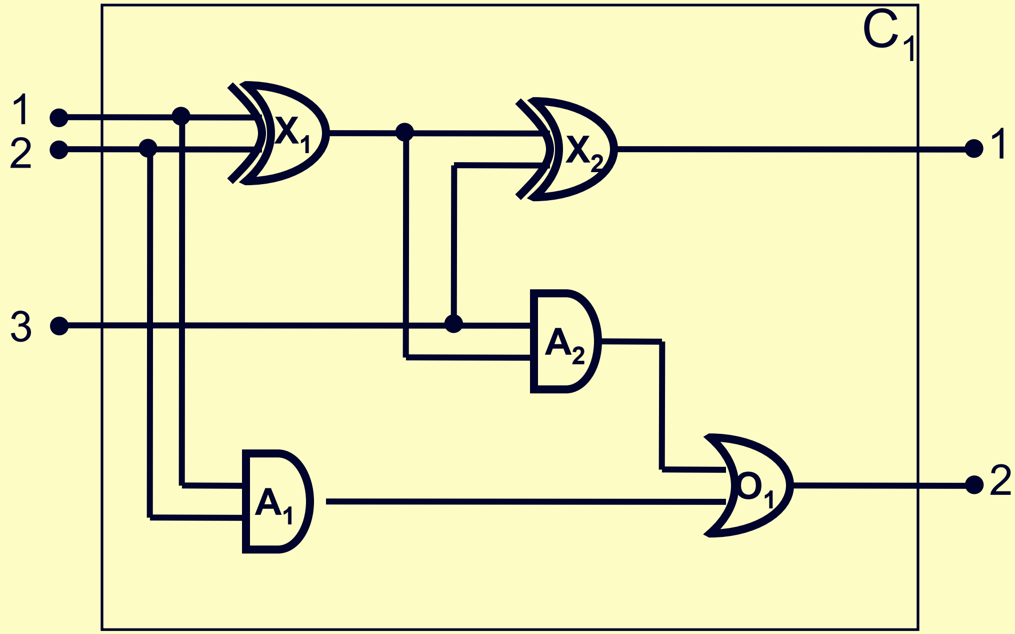
Example: Electronic Circuits (cont.)

- background knowledge
  - two connected terminals have the same signal
    - $\forall\, t_1, t_2 \quad Connected(t_1, t_2) \quad \Rightarrow Signal(t_1) = Signal(t_2)$
  - signals must be either on of off, but not both
    - $\forall\, t \quad Signal(t) = On \;\vee\; Signal(t) = Off$

      $On \neq Off$
  - connections go both ways (commutative)
    - $\forall\, t_1, t_2 \quad Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$
  - definition of *OR*
    - $\forall\, g \quad Type(g) = OR \quad \Rightarrow Signal(Out(1,g)) = On \;\Leftrightarrow\; \exists\, n\; Signal(In(n,g)) = On$
  - definition of *AND*
    - $\forall\, g \quad Type(g) = AND \quad \Rightarrow Signal(Out(1,g)) = Off \;\Leftrightarrow\; \exists\, n\; Signal(In(n,g)) = Off$
  - definition of *XOR* (sometimes denoted by ⊕)
    - $\forall\, g \quad Type(g) = XOR \quad \Rightarrow$

      $Signal(Out(1,g)) = On \quad \Leftrightarrow \quad Signal(In(1,g)) \neq Signal(In(2,g))$
  - definition of *NOT*
    - $\forall\, g \quad Type(g) = NOT \quad \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

Example: Electronic Circuits (cont.)



- specific problem: circuit C_1 to be modeled

  Type(X_1) = XOR; Type(X_2) = XOR;

  Type(A_1) = AND; Type(A_2) = AND;

  Type(O_1) = OR;

  Connected(In(1, C_1), In(1, X_1));

  Connected(In(1, C_1), In(1, A_1));

  Connected(In(2, C_1), In(2, X_1));

  Connected(In(2, C_1), In(2, A_1));

  Connected(In(3, C_1), In(2, X_2));

  Connected(In(3, C_1), In(1, A_2));

  Connected(Out(1, X_1), In(1, X_2));

  Connected(Out(1, X_1), In(2, A_2));

  Connected(Out(1, A_2), In(1, O_1));

  Connected(Out(1, A_1), In(2, O_1));

  Connected(Out(1, X_2), Out(1, C_1));

Example: Electronic Circuits (cont.)

- queries
  - When is the first output of C_1 (sum bit) off and the second one (carry bit) on?

$$\exists\, i_1, i_2, i_3\ Signal(In(1, C_1)) = i_1 \land Signal(In(2, C_1)) = i_2 \land Signal(In(3, C_1)) = i_3 \land Signal(Out(1, C_1)) = Off \land Signal(Out(2, C_1)) = On$$

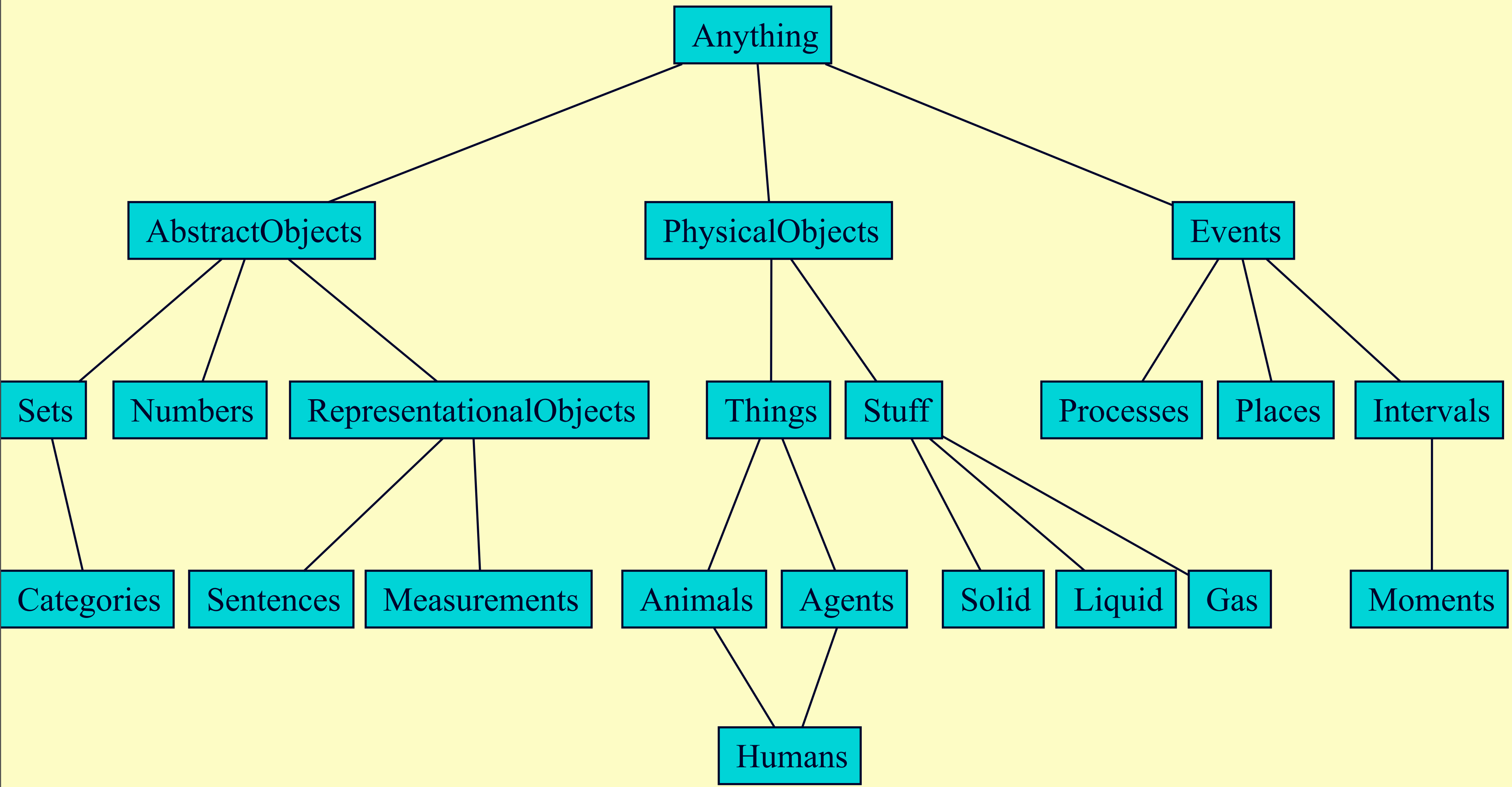
  - Give all combinations of the values for the terminals of C_1

$$\exists\, i_1, i_2, i_3, o_1, o_2\ \ Signal(In(1, C_1)) = i_1 \land Signal(In(2, C_1)) = i_2 \land Signal(In(3, C_1)) = i_3 \land Signal(Out(1, C_1)) = o_1 \land Signal(Out(2, C_1)) = o_2$$

Ontologies

- define the terminology about the objects and their relationships in a systematic way
  - closely related to taxonomies, classifications
    - ontologies don't have to be hierarchical
    - emphasis on the terms to describe objects, relationships, not on the properties of objects or specific relationships between objects
- general ontology
  - convergence of a multitude of special-purpose ontologies
  - should be applicable to any special-purpose domain

Example General Ontology



there is no generally agreed upon ontology

Issues for Ontologies

- categories
- measures
- composite objects
- time, space, and change
- events and processes
- physical objects
- substances
- mental objects and beliefs

Categories

- categories are very important for reasoning
  - almost always organized as taxonomic hierarchies
  - general statements about related objects can be made easily
  - specific properties of instances can either be inferred, or specified explicitly
  - inheritance
    - similar to OO programming

Measures

- values for properties of objects
- frequently expressed quantitatively
  - number and unit function
  - allows the use of ordering function on objects

Composite Objects

- objects frequently can be decomposed into parts, or composed into larger objects
- often expressed through *PartOf relation*
  - allows grouping of objects into hierarchies
- frequently the internal structure of objects is of importance
  - allows general reasoning about certain aspects of objects

Time, Space, and Change

- can be described around the notion of events and processes
  - events are discrete
  - processes are continuous
    - sometimes also called *liquid events*

Events

- an event is an object with temporal and spatial extent
  - it occurs somewhere for a certain duration
  - this viewpoint implies some similarities with physical objects
    - also have temporal and spatial extent
  - events may also have internal structure
- intervals are special events
  - they include all sub-events during a given time period
- places are special events
  - fixed in time, with a temporal extent

Objects

- sometimes it is useful to view some types of objects as *fluents*
  - may change during its existence, but still be considered as an object
    - e.g. the country of Germany
    - the president of the U.S.

Substances

- allows the grouping of large numbers of primitive objects into "stuff"
  - denoted by *mass nouns* in contrast to *count nouns*
  - dividing stuff into smaller pieces yields the same type of stuff
    - quantity is different
  - intrinsic properties
    - belong to the substance of an object
    - are retained under subdivisions
  - extrinsic properties
    - come from the specific object as a whole
    - change or get lost under subdivision

Mental Objects and Beliefs

- mental objects are “in one’s head”
  - allows the agent to reason about its mental processes
    - some mental processes are the reasoning processes themselves
    - the agent then can perform higher-level reasoning
    - leads to considerable technical and philosophical complications
    - assumed to be a pre-condition for consciousness
- beliefs are used to make statements about mental objects

Important Concepts and Terms

- agent
- automated reasoning
- belief
- category
- change
- composite object
- domain
- event
- hierarchy
- inference
- knowledge engineering
- knowledge representation
- logic
- measure
- mental object
- object
- ontology
- physical object
- predicate logic
- process
- propositional logic
- rule
- space
- substance
- time
- vocabulary

Chapter Summary

- knowledge representation is a fundamental aspect of reasoning systems
- knowledge representation is often done in stages
  - domain selection
  - vocabulary definition
  - encoding of background knowledge
  - specification of the problem
  - formulation of queries
- ontologies are used for capturing the terminology of a domain
- knowledge engineers act as intermediaries between users, domain experts and programmers