CS 561: Artificial Intelligence

Instructor: Sofus A. Macskassy, macskass@usc.edu
TAs: Nadeesha Ranasinghe (nadeeshr@usc.edu)
William Yeoh (wyeoh@usc.edu)
Harris Chiu (chiciu@usc.edu)

Lectures: MW 5:00-6:20pm, OHE 122 / DEN
Office hours: By appointment
Class page: http://www-rcf.usc.edu/~macskass/CS561-Spring2010/

This class will use http://www.uscden.net/ and class webpage
- Up to date information
- Lecture notes
- Relevant dates, links, etc.

Course material:
Knowledge Representation [AIMA Ch 10]

- Knowledge engineering: principles and pitfalls
- Ontologies
- Examples
Knowledge Engineer

- Populates KB with facts and relations
- Must study and understand domain to pick important objects and relationships
- **Main steps:**
  - Decide what to talk about
  - Decide on vocabulary of predicates, functions & constants
  - Encode general knowledge about domain
  - Encode description of specific problem instance
  - Pose queries to inference procedure and get answers
## Knowledge engineering vs. programming

<table>
<thead>
<tr>
<th>Knowledge Engineering</th>
<th>Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Choose a logic</td>
<td>Choose prog. language</td>
</tr>
<tr>
<td>2. Building knowledge base</td>
<td>Writing program</td>
</tr>
<tr>
<td>3. Implementing proof theory</td>
<td>Choosing/writing compiler</td>
</tr>
<tr>
<td>4. Inferring new facts</td>
<td>Running program</td>
</tr>
</tbody>
</table>

Why knowledge engineering rather than programming? **Less work:** just specify objects and relationships known to be true, but leave it to the inference engine to figure out how to solve a problem using the known facts.
Properties of good knowledge bases

- Expressive
- Concise
- Unambiguous
- Context-insensitive
- Effective
- Clear
- Correct
- ...

Trade-offs: e.g., sacrifice some correctness if it enhances brevity.
Efficiency

- **Ideally:** Not the knowledge engineer’s problem

  The inference procedure should obtain same answers no matter how knowledge is implemented.

- **In practice:**
  - use automated optimization
  - knowledge engineer should have some understanding of how inference is done
Pitfall: design KB for human readers

- KB should be designed primarily for inference procedure!
- e.g., *VeryLongName* predicates:

\[ \text{BearOfVerySmallBrain}(\text{Pooh}) \] does not allow inference procedure to infer that \textit{Pooh} is a bear, an animal, or that he has a very small brain, ...

Rather, use:

\[ \text{Bear}(\text{Pooh}) \]
\[ \forall \ b, \text{Bear}(b) \Rightarrow \text{Animal}(b) \]
\[ \forall \ a, \text{Animal}(a) \Rightarrow \text{PhysicalThing}(a) \]

...
Debugging

In principle, easier than debugging a program, because we can look at each logic sentence in isolation and tell whether it is correct.

Example:
\( \forall x, \text{Animal}(x) \Rightarrow \exists b, \text{BrainOf}(x) = b \) means “there is some object that is the value of the BrainOf function applied to an animal” and can be corrected to mean “every animal has a brain” without looking at other sentences.
Ontology

- Collection of concepts and inter-relationships
- Widely used in the database community to "translate" queries and concepts from one database to another, so that multiple databases can be used conjointly (database federation)
Ontology Example

Khan & McLeod, 2000
Towards a general ontology

Develop good representations for:

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

- We interact with individual objects, but...
  much of reasoning takes place at the level of categories.

**Representing categories in FOL:**
- use **unary predicates**
  e.g., $\text{Tomato}(x)$
- in a table form (small set of objects)
- based on its properties

- **reification**: turn a predicate or function into an object
  e.g., use constant symbol Tomatoes to refer to set of all tomatoes
  “$x$ is a tomato” expressed as “$x \in \text{Tomatoes}$”

- Strong property of reification: can make assertions about reified category itself rather than its members
  e.g., $\text{Population}(\text{Humans}) = 5e9$
Categories: inheritance

- Allow to organize and simplify knowledge base
  
  e.g., if all members of category *Food* are edible and *Fruits* is a subclass of *Food* and *Apples* is a subclass of *Fruits* then we know (through inheritance) that apples are edible.

- **Taxonomy**: hierarchy of subclasses

- Because categories are sets, we handle them as such. e.g., two categories are **disjoint** if they have no member in common. A disjoint exhaustive decomposition is called a **partition** etc...
Example: Taxonomy of hand/arm movements

- Hand/arm movement
  - Gestures
    - Manipulative
    - Communicative
      - Acts
        - Mimetic
        - Deictic
      - Symbols
        - Referential
        - Modalizing

Measures

- Can be represented using units functions
  e.g., \( \text{Length}(L_1) = \text{Inches}(1.5) = \text{Centimeters}(3.81) \)

- Measures can be used to describe objects
  e.g., \( \text{Mass}(\text{Tomato}_{12}) = \text{Kilograms}(0.16) \)

- Caution: be careful to distinguish between measures and objects
  e.g., \( \forall b, b \in \text{DollarBills} \Rightarrow \text{CashValue}(b) = $(1.00) \)
Composite Objects

- One object can be part of another.
- PartOf relation is transitive and reflexive:
  e.g., PartOf(Bucharest, Romania)
    PartOf(Romania, EasternEurope)
    PartOf(EasternEurope, Europe)
  Then we can infer Part Of(Bucharest, Europe)

- Composite object: any object that has parts
Categories of composite objects often characterized by their structure, i.e., what the parts are and how they relate.

e.g., \( \forall a \) Biped(a) \( \Rightarrow \)
\[
\exists ll, lr, b \\
\text{Leg}(ll) \land \text{Leg}(lr) \land \text{Body}(b) \land \\
\text{PartOf}(ll, a) \land \text{PartOf}(lr, a) \land \text{PartOf}(b, a) \land \\
\text{Attached}(ll, b) \land \text{Attached}(lr, b) \land \\
ll \neq lr \land \\
\forall x \text{ Leg}(x) \land \text{PartOf}(x, a) \Rightarrow (x = ll \lor x = lr)
\]

Such description can be used to describe any objects, including events. We then talk about schemas and scripts.


Events

- Chunks of spatio-temporal universe

  e.g., consider the event WorldWarII
  it has parts or sub-events: SubEvent(BattleOfBritain, WorldWarII)
  it can be a sub-event: SubEvent(WorldWarII, TwentiethCentury)

- **Intervals**: events that include as sub-events all events occurring in a given time period (thus they are temporal sections of the entire spatial universe).

- Cf. situation calculus: fact true in particular situation
  event calculus: event occurs during particular interval
Events (cont.)

- Places: spatial sections of the spatio-temporal universe that extend through time
- Use $\text{In}(x)$ to denote subevent relation between places; e.g. $\text{In}($NewYork, USA$)$

- **Location function**: maps an object to the smallest place that contains it:

  $$\forall x, l \, \text{Location}(x) = l \iff \text{At}(x, l) \land \forall l' \, \text{At}(x, l') \Rightarrow \text{In}(l, l')$$
Times, Intervals and Actions

- Time intervals can be partitioned between moments (=zero duration) and extended intervals:

- Absolute times can then be derived from defining a time scale (e.g., seconds since midnight GMT on Jan 1, 1900) and associating points on that scale with events.

- The functions Start and End then pick the earliest and latest moments in an interval. The function Duration gives the difference between end and start times.

\[ \forall i \text{ Interval}(i) \Rightarrow \text{Duration}(i) = (\text{Time(End}(i) - \text{Time(Start}(i))) \]

```
Time(Start(AD1900)) = Seconds(0)
Time(Start(AD1991)) = Seconds(2871694800)
Time(End(AD1991)) = Seconds(2903230800)
Duration(AD1991) = Seconds(31536000)
```
Then we can define predicates on intervals such as:

\[ \forall i, j \text{ Meet}(i, j) \Leftrightarrow \text{Time(End}(i)) = \text{Time(Start}(j)) \]
\[ \forall i, j \text{ Before}(i, j) \Leftrightarrow \text{Time(End}(i)) < \text{Time(Start}(j)) \]
\[ \forall i, j \text{ After}(j, i) \Leftrightarrow \text{Before}(i, j) \]
\[ \forall i, j \text{ During}(i, j) \Leftrightarrow \text{Time(Start}(j)) \leq \text{Time(Start}(i)) \land \text{Time(End}(j)) \geq \text{Time(End}(i)) \]
\[ \forall i, j \text{ Overlap}(i, j) \Leftrightarrow \exists k \text{ During}(k, i) \land \text{During}(k, j) \]
Objects Revisited

- It is legitimate to describe many objects as events.

- We can then use temporal and spatial sub-events to capture changing properties of the objects.

  e.g.,
  
  Poland event
  19thCenturyPoland temporal sub-event
  CentralPoland spatial sub-event

We call fluents objects that can change across situations.
Substances and Objects

• Some objects cannot be divided into distinct parts –
  e.g., butter: one butter? no, some butter!
  ⇒ butter substance (and similarly for temporal substances)
  (simple rule for deciding what is a substance: if you cut it in half, you should get the same).

How can we represent substances?

- Start with a category
  e.g., \( \forall x, y \ \ x \in \text{Butter} \land \text{PartOf}(y, x) \Rightarrow y \in \text{Butter} \)

- Then we can state properties
  e.g., \( \forall x \ \text{Butter}(x) \Rightarrow \text{MeltingPoint}(x, \text{Centigrade}(30)) \)
Example: Activity Recognition

- **Goal:** use network of video cameras to monitor human activity

- **Applications:** surveillance, security, reactive environments

- **Research:** IRIS at USC

- **Examples:** two persons meet, one person follows another, one person steals a bag, etc...
Human activity detection

- Nevatia/Medioni/Cohen
Low-level processing

Figure 4: Example of construction of paths from optical flow field in the $2D + t$ space.

Figure 5: Integration along a beam of paths of the motion field for robust inference of a pixel trajectory. 

a. Illustration of the beam for a circular domain $\Omega$. 

b. Illustration of the measure function $\mu(\omega)$ along the $x$—axis.
Spatio-temporal representation

Diagram showing the process of inferring the structured representation of a video stream:

1. Detected Regions
2. Structured representation
3. 2D+t Representation
4. Trajectory Extraction
5. Velocity Estimation
6. Tensor-based filtering

Figure 6: Inferring the structured representation of a video stream.
Figure 7: Structured representation of a video stream of two persons moving in a parking lot. (a) Detected moving regions, (b) $2D + t$ representation and inference of trajectories, (c) Mapping of the structured representation onto the original video frames.
Modeling Events

<table>
<thead>
<tr>
<th>Spatial Location</th>
<th>Primary Motion</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>toward / away</td>
</tr>
<tr>
<td>inside / outside</td>
<td>up / down</td>
</tr>
<tr>
<td>near / far</td>
<td>into / out of</td>
</tr>
<tr>
<td>next to</td>
<td>past</td>
</tr>
<tr>
<td>between</td>
<td>along</td>
</tr>
<tr>
<td>above / below</td>
<td></td>
</tr>
<tr>
<td>among</td>
<td>around</td>
</tr>
<tr>
<td>the front/back of</td>
<td></td>
</tr>
<tr>
<td>on top of</td>
<td>through / across</td>
</tr>
<tr>
<td>on bottom of</td>
<td>after / before</td>
</tr>
</tbody>
</table>

Table 1: English spatial prepositions (simplified from [27])

(a) Event "contact": A person approaches another person, then makes a contact, and then turns around and leaves.

(b) Event "Passing_by": A person approaches another person, then passes by, and then leaves.

Figure 12: Modeling of two similar complex, single-thread events related to the meeting pattern of two persons. Each event is composed of three simple sub-events.
Modeling Events

Figure 13: A global view of our proposed scenario modeling; scenarios are defined as a single-thread or a multi-thread event which is described by the associated mobile object properties and image features.
I) CONTACT1

(b) Recognition results of two competing activities.

II) PASSING_BY

Figure 15: (a) Input sequence A shows a complex, single thread event “Contact1”. Object 1 (at the top) approaches object 2 (at the bottom), makes contact (both objects have merged as they meet), turns around and leaves. (b) Event “Contact1” is recognized with $P(MS^*|O) = 0.7$. Event “Passing By” is recognized with lower probability (almost 0 at the end) since sub-event “leaving without turning around” is not established.