CS325 Artificial Intelligence
Chs. 9, 12 – Knowledge Representation and Inference

Cengiz Günay, Emory Univ.

LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.

I WONDER IF THE CYCLES WILL CONTINUE FOREVER.

A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.

THese ARE YOUR FATHER'S PARENTheses

ELEGANT WEAPONS

FOR A MORE... CIVILIZED AGE.

Spring 2013
Exit survey: Logic

- Where would you use propositional vs. FOL?
- What is the importance of logic representation over what we saw earlier?

Entry survey: Knowledge Representation and Inference (0.25 points of final grade)

- What is the difference between data, information and knowledge?
- What do you think would count as a “knowledge base”?
Part I: The Variable Binding Problem
Propositional Logic: Facts only
First Order Logic: Objects, variables, relations
Reminder: Propositional Logic vs. First Order Logic

Propositional Logic: Facts only
First Order Logic: Objects, variables, relations

Let’s talk about my brain research!
Single neurons can represent concepts in the brain

- Human brain only takes a second to recognize an object or a person
- How this high-level representation achieved is unknown
Single neurons can represent concepts in the brain

- Human brain only takes a second to recognize an object or a person
- How this high-level representation achieved is unknown
- But can find single neurons representing, e.g., actress Jennifer Aniston:

Quiroga et al. (2005)
... even when it is an abstraction

- These neurons also respond to abstract notions of the same concept (e.g., actress Halle Berry):

Quiroga et al. (2005)
Then, are features always represented by single neurons?

The Binding Problem (1)

Rosenblatt’s example (1961): two shapes in two possible locations in a visual scene.
Objects can be detected individually, but not when together.

If propositional representations are employed:

- triangle-object $\wedge$ object-in-upper-part
- square-object $\wedge$ triangle-object
  $\wedge$
- object-in-upper-part $\wedge$ object-in-lower-part

Both satisfies query: triangle-object $\wedge$ object-in-upper-part $\Rightarrow$ something
An LTU neuron suffers from this binding problem.

This linear threshold unit (LTU) neuron exhibits the same problem:

Inputs:
- square
- triangle
- upper
- lower

Linear Threshold Unit

Output:
- foo

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Linear Threshold Unit</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>square</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>triangle</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>upper</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>lower</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Possible Solution (1): Combination-coding

Using a neuron for each possible configuration combination, i.e.:
- upper-triangle, upper-square, lower-triangle, lower-square.

Drawback: Combinatorial explosion:
- Impossible that the brain has individual cells for each possible concept combination in nature (Barlow, 1972).
Possible Solution (2): Phase-coding with Temporal Binding

Bound entities are represented by temporal synchrony:

Query $\triangle \land \text{upper} \Rightarrow$ something is only satisfied by the top case!
Temporal binding:

- Recent evidence of binding units in monkeys (Stark et al., 2008)
- But, only allows temporary representations (O’Reilly et al., 2003)

**Recruitment learning** (Feldman, 1982; Diederich, Günay & Hogan, 2010) forms long-term memories, which:

- Can be induced by temporal binding (Valiant, 1994; Shastri, 2001);
- Models the brain as a random graph (Wickelgren, 1979).
- Avoids combinatorial explosion by only allocating when needed (Feldman, 1990; Valiant, 1994; Page, 2000).
Brain Uses Time to Encode Variables?

- Still a valid theory
- We don’t know how the brain represents binding information
- Other theories: synfire chains, synchronized oscillations
Part II: Inference
Automated Inference?

We already did it:

What we know to be True:

- \((E \lor B) \Rightarrow A\)
- \(A \Rightarrow (J \land M)\)
- \(B\)

Can we infer? T F ?

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>
Automated Inference?

We already did it:

What we know to be True:

- \((E \lor B) \Rightarrow A\)
- \(A \Rightarrow (J \land M)\)
- \(B\)

Can we infer? 

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>J</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>M</td>
</tr>
</tbody>
</table>
Automated Inference?

We already did it:

What we know to be True:

- \((E \lor B) \Rightarrow A\)
- \(A \Rightarrow (J \land M)\)
- \(B\)

Can we infer? T F ?

In propositional logic, resolution by forward/backward chaining

**Forward:** Start from knowledge to reach query

**Backward:** Start from query and go back
Automated Inference?

We already did it:

What we know to be True:

- \((E \lor B) \Rightarrow A\)
- \(A \Rightarrow (J \land M)\)
- \(B\)

Can we infer? T F ?

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>J</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>M</td>
</tr>
</tbody>
</table>

- In propositional logic, resolution by forward/backward chaining
  - **Forward**: Start from knowledge to reach query
  - **Backward**: Start from query and go back
- In FOL, substitute variables to get propositions (see Ch. 9)
  - Use lifting and unification to resolve variables
- Logic programming: Prolog, LISP, Haskell
Most widely used logic language.

Rules are written in backwards:
  criminal(X) :- american(X), weapon(Y), sells(X, Y, Z), hostile(Z)

Variables are uppercase and constants lowercase.
Prolog

- Most widely used logic language.
- Rules are written in backwards:
  
  \[
  \text{criminal} \,(X) \,\text{:-}\, \text{american}(X), \text{weapon}(Y), \text{sells} \,(X,\,Y,\,Z), \,\text{hostile}\,(Z)
  \]

- Variables are uppercase and constants lowercase.

- Because of complexity, often compiled into other languages like: Warren Abstract Machine, LISP or C.

- Language makes it easy to construct lists, like LISP.
LISP  LISPt Processing language: primary data structure is lists.
LISP  LIST Processing language: primary data structure is lists.

- Lisp is used for AI because it can work with symbols.
- Examples: computer algebra, theorem proving, planning systems, diagnosis, rewrite systems, knowledge representation and reasoning, logic languages, machine translation, expert systems, ...
- It is a *functional* programming language, as opposed to a *procedural* or *imperative* language.
Functional languages

- LISP invented by John McCarthy in 1958

(defun factorial (n)
  (if (<= n 1)
      1
      (* n (factorial (- n 1))))
Functional languages

- LISP invented by John McCarthy in 1958
- Scheme: A minimalist LISP since 1975. Introduces \textit{lambda calculus}.

\[(\text{define-syntax let}
\begin{align*}
&\text{(syntax-rules ()}
&\begin{align*}
&\text{((let ((var expr) \ldots) body \ldots) }
&\text{((lambda (var \ldots) body \ldots) expr \ldots)))}
\end{align*}
\end{align*}\]
Functional languages

- LISP invented by John McCarthy in 1958
- Scheme: A minimalist LISP since 1975. Introduces *lambda calculus*.

```
(define-syntax let
  (syntax-rules ()
    ((let ((var expr) ...) body ...)
      ((lambda (var ...) body ...) expr ...)))
```

Java implementation JScheme by Peter Norvig in 1998.

```
java jscheme.Scheme scheme-files...
```
Functional languages

- LISP invented by John McCarthy in 1958
- Scheme: Since 1975. Introduces \textit{lambda calculus}.
- Haskell: Lazy functional language in 90s.

\begin{verbatim}
\textit{Type annotation (optional)}
factorial :: \textbf{Integer} \rightarrow \textbf{Integer}

\textit{Using recursion}
factorial 0 = 1
factorial n = n * factorial (n - 1)
\end{verbatim}
Famous AI applications in Lisp:

- Macsyma as the first large computer algebra system.
- ACL2 as a widely used theorem prover, for example used by AMD.
- DART as the logistics planner used during the first Gulf war by the US military. This Lisp application alone is said to have paid back for all US investments in AI research at that time.
- SPIKE, the planning and scheduling application for the Hubble Space Telescope. Also used by several other large telescopes.
- CYC, one of the largest software systems written. Representation and reasoning in the domain of human common sense knowledge.
- METAL, one of the first commercially used natural language translation systems.
- American Express’ Authorizer’s Assistant, which checks credit card transactions.
So What’s So Special About LISP?

- First language to be **homoiconic**: data and code represented alike, can modify and execute code on the fly
So What’s So Special About LISP?

- **First language to be homoiconic**: data and code represented alike, can modify and execute code on the fly
  - Ever used Java introspection? Scripting languages like PERL and Python allow evaluating new code, too.
So What’s So Special About LISP?

- First language to be **homoiconic**: data and code represented alike, can modify and execute code on the fly
  - Ever used Java introspection? Scripting languages like PERL and Python allow evaluating new code, too.
- First use of the *if-then-else* structure
- Adopted object-oriented features from language *SmallTalk*
- First use of *automatic garbage collection*
Part III: Knowledge Representation (Ch. 12)
The Open Biological and Biomedical Ontologies

The OBO Foundry is a collaborative experiment involving developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of orthogonal interoperable reference ontologies in the biomedical domain. The groups developing ontologies who have expressed an interest in this goal are listed below, followed by other relevant efforts in this domain.

In addition to a listing of OBO ontologies, this site also provides a statement of the OBO Foundry principles, discussion fora, technical infrastructure, and other services to facilitate ontology development. We welcome feedback and encourage participation.

Click any column header to sort the table by that column. The footnotes link to the term request trackers for the listed ontologies.

<table>
<thead>
<tr>
<th>OBO Foundry ontologies</th>
<th>Title</th>
<th>Domain</th>
<th>Prefix</th>
<th>File</th>
<th>Last changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biological process</td>
<td>biological process</td>
<td>GO</td>
<td>go.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cellular component</td>
<td>anatomy</td>
<td>GO</td>
<td>go.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chemical entities of biological interest</td>
<td>biochemistry</td>
<td>CHEBI</td>
<td>chebi.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Molecular function</td>
<td>biological function</td>
<td>GO</td>
<td>go.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phenotypic quality</td>
<td>phenotype</td>
<td>PATO</td>
<td>quality.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRotein Ontology (PRO)</td>
<td>proteins</td>
<td>PR</td>
<td>pro.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xenopus anatomy and development</td>
<td>anatomy</td>
<td>XAO</td>
<td>xenopus_anatomy_edit.obo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zebrafish anatomy and development</td>
<td>anatomy</td>
<td>ZFA</td>
<td>zebrafish_anatomy.obo</td>
<td>2013/02/07</td>
<td></td>
</tr>
</tbody>
</table>
Knowledge Ontologies

format-version: 1.2
data-version: 2013-02-14
date: 13:02:2013 16:50
saved-by: tanyaberardini
auto-generated-by: OBO-Epit 2.3
subsetdef: Cross_product_review "Involved_in"
subsetdef: goslim_aspergillus "Aspergillus GO slim"
subsetdef: goslim_candida "Candida GO slim"
subsetdef: goslim_generic "Generic GO slim"
subsetdef: goslim_metagenomics "Metagenomics GO slim"
subsetdef: goslim_pir "PIR GO slim"
subsetdef: goslim_plant "Plant GO slim"
subsetdef: goslim_pombe "Fission yeast GO slim"
subsetdef: goslim_yeast "Yeast GO slim"
subsetdef: gosubset_prok "Prokaryotic GO subset"
subsetdef: high_level_annotation_qc "High-level terms not to be used for direct annotation"
subsetdef: mf_needs_review "CataLytic activity terms in need of attention"
subsetdef: termgenie_unvetted "Terms created by TermGenie that do not follow a template and req
subsetdef: virus_checked "Viral overhaul terms"
synonym typedef: systematic_synonym "Systematic synonym" EXACT
default-namespace: gene_ontology
remark: cvs version: $Revision: 6348 $
ontology: go

[Term]
id: GO:0000001
name: mitochondrion inheritance
namespace: biological_process
def: "The distribution of mitochondria, including the mitochondrial genome, into daughter cells
synonym: "mitochondrial inheritance" EXACT []
is_a: GO:0048308 ! organelle inheritance
is_a: GO:0048311 ! mitochondrion distribution

[Term]
id: GO:0000002
name: mitochondrial genome maintenance
namespace: biological_process
def: "The maintenance of the structure and integrity of the mitochondrial genome; includes repl
is_a: GO:0007005 ! mitochondrion organization
How to Define an Ontology?

Ontological language must represent:

**Categories:** Groups

**Composition:** \( \text{PartOf}(\text{Bucharest, Romania}) \)
Can define hierarchical **taxonomy**

**Relations:** Between objects

**Events, Processes:** \( \text{Happens}(e, i) \)

**Quantities:** \( \text{Centimeters}(3.81) \)
(Continuous values are problematic)

**Time:** \( \text{Interval}(i), \text{Time}(\text{Begin}(1987)) \)
Never took off, better write it with **description logic**
Standards for machine readable knowledge for the web exist:

**OWL**: Description logic

**RDP**: Relational logic
Standards for machine readable knowledge for the web exist:

**OWL**: Description logic

**RDP**: Relational logic

- But they are not widely used (except for knowledge bases)
- Other web agents are emerging
Web Agents

- Crawlers, IFTTT, Yahoo pipes

**What is IFTTT?**
IFTTT is a service that lets you create powerful connections with one simple statement:

```
if this then that
```

IFTTT is pronounced like "gift" without the "g."

**Channels**
Channels are the basic building blocks of IFTTT. Each Channel has its own Triggers and Actions. Some example Channels are:

- Facebook
- Evernote
- Email
- Weather
- LinkedIn

View all 59 Channels

**Triggers**
The *this* part of a Recipe is a Trigger. Some example Triggers are "I'm tagged in a photo on Facebook" or "I check in on Foursquare."

**Actions**
The *that* part of a Recipe is an Action. Some example Actions are "send me a text message" or "create a status message on Facebook."
Crawlers, IFTTT, Yahoo pipes
Web Agents

- Crawlers, IFTTT, Yahoo pipes


