Limitations of Propositional Logic => Predicate Calculus
Announcements

- New homeworks related to the projects posted
- Update your project proposals, due Friday
- Midterm Feb 23rd
- Pop Quiz key discussion, return proposals and pop quizzes after class.
Key: Pop Quiz

- List the time(s) your team has arranged to meet every week.
  - Specific TIMES set aside to meet each week. Specific RESPONSIBILITIES for each team member.

- Name the modules that a dialogue system must typically have
  - NLU, DM, Database, NLG (some people actually didn’t know this)

- Describe in one sentence the input and the output for each of these modules
  - NLU: words => semantic representation (e.g. dialogue act + arguments)
  - DM: semantic representation to database query, database tuples to specification of dialogue response to give to NLG
  - Database: database query => database tuples
  - NLG: semantic representation (content plan) => output string
Describe in up to 10 sentences why you were asked to produce a corpus of utterances for your bot, how you did that, and what you will use it for.

- We are trying to make it possible to test each module separately by constructing sets of inputs/outputs for each module.
- The NLU was the first module, we were supposed to construct strings for inputs and specify the semantic representation for outputs.
- We decided to use domain specific dialogue act tagging and chunking as our techniques for producing the semantic output.
- Given our corpus of NLU translated outputs, we will start testing our dialogue manager.

Give a list of three things that might count as features/capabilities that would make your system count as intelligent.

List one or more good performance metrics that your team intends to use to evaluate your BOT as a whole, and say WHY.
Key: Pop Quiz (continued)

- Give a list of three things that might count as features/capabilities that would make your system count as intelligent
  - Our system is going to be able to resolve pronouns (he, she, it) and ellipsis (how about with Julia Roberts)
  - Our system will make inferences using an ontology so it can generalize from the question that was asked.
  - Our system will use Wordnet in the NLG component so it can do interesting word selection depending on how smart it thinks the user is, or how old.
  - Our WOW system is going to use intelligent path planning, and rank routes we suggest to the user by how safe they are or how likely they are to succeed.
  - Our system is going to develop a user model to use to rank outputs from the DB and modify this model based on the history of interaction with a particular user.
  - Our system is going to have two modules for NLU, one is going to be rule-based and the other one will be trained from corpora. We hope to be able to show that unless you have good training methods and enough data the trained version can’t outperform the rule-based one, but they make errors on different inputs. Therefore we hypothesize an interesting thing to try might be to use them together by ‘voting’
Key: Pop Quiz (continued)

- Give a list of three things that might count as features/capabilities that would make your system count as intelligent
  - We are going to design our dialogue manager to be trainable using reinforcement learning. We will produce a very simple user simulation to debug the training method. Then we plan to use ourselves as subjects with user satisfaction as the feedback metric to show that we can train the system to improve its performance over time.
  - Our system is going to respond to ‘indirect speech acts’, e.g. “I wonder who won an Oscar in 2000.”
  - Our system is going to keep track of recipes you liked or didn’t like.
  - Our system is going to keep track of what you’ve told it you have in the liquor cabinet or cupboard.
  - Our system is going to have personality and modify the way it expresses itself for different users. We are going to do this with simple indexing on different ways to say things and by asking the user about their personality. But we hope to be able to show that users like the system better if it matches their personality. Or at least that users like some personalities better than others.
List one or more good performance metrics that your team intends to use to evaluate your BOT as a whole, and say WHY.
- This is the P of the PEAS model for dialogue agents
- These were covered in the lecture of Jan 26\textsuperscript{th}.

**Task Success**
- Ask user ‘were you successful’
- Measure matching attributes for fixed tasks
- Ask user to rank different paths returned for WOW and see whether user ranking matches your ranking

**User Satisfaction (from a survey)**

**Concept Accuracy (for NLU)**

**Dialogue Act classification accuracy (for NLU)**

**Does user like the Pirate personality of NLG or prefers a more boring personality for the agent?**
System (Extrinsic) vs. Component (Intrinsic) Evaluation (1/25)

- PEAS Performance: performance on the target task (also called system, black-box, extrinsic evaluation)
- What is responsible for the overall performance? =>
- Research/Science on individual modules:
  - How accurate is your NLU?
  - Does your NLG convey emotion or personality?
- Plug & Play: define module interfaces such that you can switch ‘simpler’ versions of modules for ‘intelligent’ ones
- Ablation Study: Knock out particular functionality to show difference in performance w/out it.
First Order Logic <=> Predicate Calculus
NLTK Inference and Prover interfaces

- NLTK-inference.html
<table>
<thead>
<tr>
<th>Name</th>
<th>Last modified</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>agents.html</td>
<td>18-Jul-2005 17:36</td>
<td>32K</td>
</tr>
<tr>
<td>agents.py</td>
<td>12-Nov-2004 02:06</td>
<td>20K</td>
</tr>
<tr>
<td>csp.html</td>
<td>18-Jul-2005 17:36</td>
<td>28K</td>
</tr>
<tr>
<td>csp.py</td>
<td>09-Dec-2004 14:55</td>
<td>19K</td>
</tr>
<tr>
<td>csp.txt</td>
<td>11-Nov-2004 16:19</td>
<td>702</td>
</tr>
<tr>
<td>doctests.html</td>
<td>18-Jul-2005 17:36</td>
<td>3.9K</td>
</tr>
<tr>
<td>doctests.py</td>
<td>03-Feb-2005 10:45</td>
<td>1.4K</td>
</tr>
<tr>
<td>doctests.txt</td>
<td>11-Nov-2004 15:23</td>
<td>513</td>
</tr>
<tr>
<td>games.html</td>
<td>18-Jul-2005 17:36</td>
<td>16K</td>
</tr>
<tr>
<td>games.py</td>
<td>12-Nov-2004 02:06</td>
<td>9.8K</td>
</tr>
<tr>
<td>guidelines.html</td>
<td>18-Jul-2005 17:36</td>
<td>9.9K</td>
</tr>
<tr>
<td>learning.html</td>
<td>18-Jul-2005 17:36</td>
<td>35K</td>
</tr>
<tr>
<td>learning.py</td>
<td>18-Nov-2004 07:07</td>
<td>23K</td>
</tr>
<tr>
<td>logic.html</td>
<td>18-Jul-2005 17:36</td>
<td>48K</td>
</tr>
<tr>
<td>logic.py</td>
<td>09-Dec-2004 14:07</td>
<td>32K</td>
</tr>
<tr>
<td>logic.txt</td>
<td>11-Nov-2004 16:27</td>
<td>1.5K</td>
</tr>
<tr>
<td>mdp.html</td>
<td>18-Jul-2005 17:36</td>
<td>8.9K</td>
</tr>
<tr>
<td>mdp.py</td>
<td>09-Dec-2004 14:17</td>
<td>5.7K</td>
</tr>
<tr>
<td>mdp.txt</td>
<td>09-Dec-2004 14:54</td>
<td>1.0K</td>
</tr>
<tr>
<td>nlp.html</td>
<td>18-Jul-2005 17:36</td>
<td>11K</td>
</tr>
<tr>
<td>nlp.py</td>
<td>05-Nov-2004 09:56</td>
<td>5.9K</td>
</tr>
<tr>
<td>nlp.txt</td>
<td>12-Nov-2004 01:22</td>
<td>1.6K</td>
</tr>
<tr>
<td>planning.html</td>
<td>18-Jul-2005 17:36</td>
<td>2.0K</td>
</tr>
<tr>
<td>planning.py</td>
<td>31-Oct-2003 01:41</td>
<td>149</td>
</tr>
<tr>
<td>probability.html</td>
<td>18-Jul-2005 17:36</td>
<td>9.3K</td>
</tr>
<tr>
<td>probability.py</td>
<td>06-Nov-2004 03:41</td>
<td>5.3K</td>
</tr>
</tbody>
</table>
AIMA Python file: logic.py:

Representations and Inference for Logic (Chapters 7-10)

- Four important data types:
  - KB: Abstract class holds a knowledge base of logical expressions
  - KB_Agent: Abstract class subclasses agents.Agent
  - Expr: A logical expression
  - Substitution: Implemented as a dictionary of var:value pairs, \{x:1, y:x\}

- Functions for doing logical inference:
  - `pl_true`: Evaluate a propositional logical sentence in a model
  - `tt_entail`: Say if a statement is entailed by a KB
  - `pl_resolution`: Do resolution on propositional sentences
  - `dpll_satisfiable`: See if a propositional sentence is satisfiable
  - `to_cnf`: Convert to conjunctive normal form
  - Unify: Do unification of two FOL sentences
  - `diff`, `simp`: Symbolic differentiation and simplification
Predicate Calculus

- Propositional Logic doesn’t allow you to access the components of an assertion:
- “it rained on Tuesday”
- But not \(\text{weather}(\text{Tuesday},\text{rain})\)
- or for all \(X\) such that \(X\) is a day of the week - \(\text{weather}(X,\text{rain})\)
Predicate Calculus definitions

- Symbols – true, false
- Constant Symbols which name objects or properties in the world, e.g. AI course, tree, tall, blue
- Variable Symbols designate general classes of objects, properties in the world
- Function Symbols map functions to constant:
  - e.g. father(X) maps each person to their unique father
- Predicate Symbols likes(John, cheese) returns T, F
- Relations friendof(X) returns set of friends.
Predicate Calculus Operators

- The connectives: AND, OR, NOT, IMPLIES, EQUALS (same as propositional logic)
- Quantifiers —
  - Universal quantifier: for_all, ∀
  - Existential quantifier: there_exists, ∃

Universal Quantifier
- Sentence true for all values of the variable
- for_all X likes(X, icecream)

Existential Quantifier
- Sentence is true for AT LEAST one value in the domain
- there_exists Y friends(Y, Peter)
Some examples

- If it doesn’t rain on Monday, Tom will go to the mountains
  - NOT weather(rain,Monday) IMPLIES go(Tom,mountains)

- Emma is a doberman pinscher and a good dog
  - good_dog(Emma) AND is_a(Emma,doberman)
Some examples

- NOT there_exists X (person(X) AND likes(X, war))
  Nobody likes war.

- there_exists X (trash(X) AND empty(right_hand) IMPLIES pick_up(X, right_hand))
  If you see trash, and your hand is empty,
  pick it up.
First Order Predicate Calculus

- The basis of almost all knowledge representation and reasoning in every area of symbolic AI
  - Natural Language Processing
  - Planning, Reasoning
  - Rule Induction in Machine Learning
  - Expert Systems
First-order logic

- Whereas propositional logic assumes the world contains facts,
- first-order logic (like natural language) assumes the world contains
  - **Objects**: people, houses, numbers, colors, baseball games, wars, …
  - **Relations**: red, round, prime, brother of, bigger than, part of, comes between, …
  - **Functions**: father of, best friend, one more than, plus, …
Syntax of FOL: Basic elements

- Constants: King_John, 2, UCSC, ...
- Predicates: Brother, >, ...
- Functions: Sqrt, Left_Leg_Of, ...
- Variables: x, y, a, b, ...
- Connectives: ¬, ⇒, ∧, ∨, ⇔
- Equality: =
- Quantifiers: ∀, ∃
Atomic sentences

Atomic sentence = \( \text{predicate} \ (term_1, ..., term_n) \)
or \( term_1 = term_2 \)

Term = \( \text{function} \ (term_1, ..., term_n) \)
or constant or variable

- \( \text{Brother(KingJohn, RichardTheLionheart)} \)
- \( \text{Length(LeftLegOf(Richard))} \)
- \( \text{Length(LeftLegOf(KingJohn))} \)
Complex sentences

- Complex sentences are made from atomic sentences using connectives

\[-S, S_1 \land S_2, S_1 \lor S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2,\]

E.g. $\text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn})$

$>(1,2) \lor \leq (1,2)$

$>(1,2) \land \neg > (1,2)$
Using FOL

The kinship domain:

- Brothers are siblings
  \[ \forall x,y \; \text{Brother}(x,y) \iff \text{Sibling}(x,y) \]
- One's mother is one's female parent
  \[ \forall m,c \; \text{Mother}(c) = m \iff (\text{Female}(m) \land \text{Parent}(m,c)) \]
- “Sibling” is symmetric
  \[ \forall x,y \; \text{Sibling}(x,y) \iff \text{Sibling}(y,x) \]
“The departure of Mr. Whitelaw from N. Ireland at this time has amazed Irish political leaders. While there was no official comment in Dublin, it would appear that the Government was not informed in advance of MR. WHITELAW’S MOVE.”

\[
\text{Departure}(X, \text{fromplace}, \text{toplace}) \Rightarrow \text{Move}(X, \text{toplace})
\]
What is required of a knowledge representation language?

- Representational adequacy: It should allow you to represent all the knowledge you need to reason with.
- Inferential adequacy: It should allow new knowledge to be inferred from a basic set of facts.
Truth in first-order logic

- Sentences are true with respect to a model and an interpretation.

- Model contains objects (domain elements) and relations among them.

- Interpretation specifies referents for:
  - Constant symbols $\rightarrow$ objects
  - Predicate symbols $\rightarrow$ relations
  - Function symbols $\rightarrow$ functional relations

- An atomic sentence $\text{predicate}(\text{term}_1, \ldots, \text{term}_n)$ is true iff the objects referred to by $\text{term}_1, \ldots, \text{term}_n$ are in the relation referred to by $\text{predicate}$.
Models for FOL: Example
Universal quantification

- $\forall <\text{variables}> <\text{sentence}>$

Everyone at UCSC is smart:
$\forall x \text{At}(x, \text{UCSC}) \Rightarrow \text{Smart}(x)$

- $\forall x P$ is true in a model $m$ iff $P$ is true with $x$ being each possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of $P$

  $\text{At}(\text{KingJohn}, \text{UCSC}) \Rightarrow \text{Smart}(\text{KingJohn})$
  $\land \text{At}(\text{Richard}, \text{UCSC}) \Rightarrow \text{Smart}(\text{Richard})$
  $\land \text{At}(\text{UCSC}, \text{UCSC}) \Rightarrow \text{Smart}(\text{UCSC})$
  $\land ...$
A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$

- Common mistake: using $\land$ as the main connective with $\forall$:
  $$\forall x \text{ At}(x, \text{ UCSC}) \land \text{Smart}(x)$$
  means “Everyone is at UCSC and everyone is smart”
Existential quantification

- \( \exists <variables> <sentence> \)

- Someone at UCSC is smart:
  - \( \exists x \ At(x, UCSC) \land Smart(x) \)

- \( \exists x P \) is true in a model \( m \) iff \( P \) is true with \( x \) being some possible object in the model

- Roughly speaking, equivalent to the **disjunction of instantiations** of \( P \)

  \[
  \begin{align*}
  & At(KingJohn, UCSC) \land Smart(KingJohn) \\
  & \lor \ At(Richard, UCSC) \land Smart(Richard) \\
  & \lor \ At(UCSC, UCSC) \land Smart(UCSC) \\
  & \lor \ ...
  \end{align*}
  \]
Another common mistake to avoid

- Typically, $\land$ is the main connective with $\exists$

- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

  $\exists x \; \text{At}(x, \text{UCSC}) \Rightarrow \text{Smart}(x)$

  is true if there is anyone who is not at UCSC!
Properties of quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$

- $\exists x \exists y$ is the same as $\exists y \exists x$
- $\exists x \forall y$ is **not** the same as $\forall y \exists x$
- $\exists x \forall y \text{ Loves}(x,y)$
  - “There is a person who loves everyone in the world”

- $\forall y \exists x \text{ Loves}(x,y)$
  - “Everyone in the world is loved by at least one person”

**Quantifier duality:** each can be expressed using the other

- $\forall x \text{ Likes}(x,\text{IceCream})$  $\iff$  $\neg \exists x \neg \text{ Likes}(x,\text{IceCream})$
- $\exists x \text{ Likes}(x,\text{Broccoli})$  $\iff$  $\neg \forall x \neg \text{ Likes}(x,\text{Broccoli})$
Equality

- \( \text{term}_1 = \text{term}_2 \) is true under a given interpretation if and only if \( \text{term}_1 \) and \( \text{term}_2 \) refer to the same object

- E.g., definition of \textit{Sibling} in terms of \textit{Parent}:

\[
\forall x, y \ Sibling(x,y) \iff \neg (x = y) \land \exists m, f \neg (m = f) \land Parent(m, x) \\
\land Parent(f, x) \land Parent(m, y) \land Parent(f, y)
\]
Using FOL

The kinship domain:

- Brothers are siblings
  \[ \forall x, y \ Brother(x, y) \iff Sibling(x, y) \]

- One's mother is one's female parent
  \[ \forall m, c \ Mother(c) = m \iff (Female(m) \land Parent(m, c)) \]

- "Sibling" is symmetric
  \[ \forall x, y \ Sibling(x, y) \iff Sibling(y, x) \]
Interacting with FOL KBs

- Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at $t=5$:

  \[
  \text{Tell}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5))
  \\
  \text{Ask}(KB, \exists a \text{ BestAction}(a, 5))
  \]

- I.e., does the KB entail some best action at $t=5$?

- Answer: Yes, \{a/Shoot\} ← substitution (binding list)

- Given a sentence $S$ and a substitution $\sigma$,
- $S\sigma$ denotes the result of plugging $\sigma$ into $S$; e.g.,
  \[
  S = \text{Smarter}(x,y) \\
  \sigma = \{x/\text{Hillary}, y/\text{Bill}\} \\
  S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})
  \]

- \text{Ask}(KB,S) returns some/all $\sigma$ such that $KB \models \sigma$
Knowledge base for the wumpus world

- **Perception**
  - $\forall t, s, b \text{ Percept}([s, b, \text{Glitter}], t) \Rightarrow \text{Glitter}(t)$

- **Reflex**
  - $\forall t \text{ Glitter}(t) \Rightarrow \text{BestAction(Grab, } t)$
Deducing hidden properties

- $\forall x, y, a, b \ Adjacent([x, y], [a, b]) \iff [a, b] \in \{[x + 1, y], [x - 1, y], [x, y + 1], [x, y - 1]\}$

Properties of squares:
- $\forall s, t \ At(Agent, s, t) \land Breeze(t) \Rightarrow Breezy(s)$

Squares are breezy near a pit:
- **Diagnostic** rule---infer cause from effect
  $\forall s \ Breezy(s) \Rightarrow [\exists r \ Adjacent(r, s) \land Pit(r)]$

  - **Causal** rule---infer effect from cause
  $\forall r \ Pit(r) \Rightarrow [\forall s \ Adjacent(r, s) \Rightarrow Breezy(s)]$
Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
4. Encode general knowledge about the domain
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base
Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power: sufficient to define wumpus world
Ontological Commitment of Logics

- **Ontological commitment** – what entities, relationships, and facts exist in world and can be reasoned about
- **Epistemic commitment** – what agents can know about the world
Database Semantics 8.2.8

- **Unique Names Assumption:**
  - Every constant refers to a distinct object

- **Closed World Assumption**
  - Atomic Sentences not known to be true are false

- **Domain Closure:**
  - There are no domain elements other than those named by the constant symbols.
Logic programming: Prolog

- Algorithm = Logic + Control

- Basis: backward chaining with Horn clauses + bells & whistles
  Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques ⇒ 60 million LIPS

- Program = set of clauses = head :- literal₁, ... literalₙ.

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
```

- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., X is Y*Z+3
- Built-in predicates that have side effects (e.g., input and output

- predicates, assert/retract predicates)
- Closed-world assumption ("negation as failure")
  - e.g., given alive(X) :- not dead(X).
  - alive(joe) succeeds if dead(joe) fails
Inference in first-order logic
Outline

- Reducing first-order inference to propositional inference
- Unification
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution
Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:
  \[ \forall v \alpha \]
  \[ \text{Subst}\{\{v/g\}, \alpha\} \]
  for any variable \( v \) and ground term \( g \)

- E.g., \( \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \) yields:
  
  \begin{align*}
  &\text{King}(John) \land \text{Greedy}(John) \Rightarrow \text{Evil}(John) \\
  &\text{King}(Richard) \land \text{Greedy}(Richard) \Rightarrow \text{Evil}(Richard) \\
  &\text{King}(\text{Father}(John)) \land \text{Greedy}(\text{Father}(John)) \Rightarrow \text{Evil}(\text{Father}(John))
  \end{align*}
Existential instantiation (EI)

- For any sentence $\alpha$, variable $v$, and constant symbol $k$ that does not appear elsewhere in the knowledge base:

$$
\exists v \alpha \\
\text{Subst}\{\{v/k\}, \alpha\}
$$

- E.g., $\exists x \ Crown(x) \land OnHead(x, John)$ yields:

$$
\text{Crown}(C_1) \land OnHead(C_1, John)
$$

provided $C_1$ is a new constant symbol, called a Skolem constant
Reduction to propositional inference

Suppose the KB contains just the following:

\[
\forall x \text{ King}(x) \land \text{ Greedy}(x) \implies \text{ Evil}(x)
\]
\[
\text{King}(\text{John})
\]
\[
\text{Greedy}(\text{John})
\]
\[
\text{Brother}(\text{Richard},\text{John})
\]

- Instantiating the universal sentence in all possible ways, we have:
  \[
  \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \implies \text{Evil}(\text{John})
  \]
  \[
  \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \implies \text{Evil}(\text{Richard})
  \]
  \[
  \text{King}(\text{John})
  \]
  \[
  \text{Greedy}(\text{John})
  \]
  \[
  \text{Brother}(\text{Richard},\text{John})
  \]

- The new KB is propositionalized: proposition symbols are

  \[
  \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Evil}(\text{John}), \text{King}(\text{Richard}), \text{etc.}
  \]
Reduction of FOL to PL.

- Every FOL KB can be propositionalized so as to preserve entailment

- (A ground sentence is entailed by new KB iff entailed by original KB)

- Idea: propositionalize KB and query, apply resolution, return result

- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., Father(Father(Father(John)))
Reduction contd.

Theorem: Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a \textit{finite} subset of the propositionalized KB

Idea: For $n = 0$ to $\infty$ do
- create a propositional KB by instantiating with depth-$n$ terms
- see if $\alpha$ is entailed by this KB

Problem: works if $\alpha$ is entailed, loops if $\alpha$ is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is \textit{semidecidable} (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)
The DPLL algorithm

EVERY FOL KB can be converted to a PL KB

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. **Early termination**
   A clause is true if any literal is true.
   A sentence is false if any clause is false.

2. **Pure symbol heuristic**
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses ($A \lor \neg B$), $(\neg B \lor \neg C)$, $(C \lor A)$, $A$ and $B$ are pure, $C$ is impure.
   Make a pure symbol literal true.

3. **Unit clause heuristic**
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.
Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.

- E.g., from:

  \[
  \forall x \text{ King}(x) \land \text{ Greedy}(x) \Rightarrow \text{ Evil}(x)
  \]

  \[
  \text{King}(\text{John})
  \]

  \[
  \forall y \text{ Greedy}(y)
  \]

  \[
  \text{Brother}(\text{Richard}, \text{John})
  \]

- it seems obvious that \(\text{Evil}(\text{John})\), but propositionalization produces lots of facts such as \(\text{Greedy}(\text{Richard})\) that are irrelevant

- With \(p\) \(k\)-ary predicates and \(n\) constants, there are \(p \cdot n^k\) instantiations.
Unification (Used heavily in NLP)

- We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha \theta = \beta \theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows(John, Jane)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($y$, OJ)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($y$, Mother($y$))</td>
<td></td>
</tr>
<tr>
<td>Knows(John, $x$)</td>
<td>Knows($x$, OJ)</td>
<td></td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
Unification

- We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$

$\theta = \{x/John, y/John\}$ works

- Unify($\alpha, \beta$) = $\theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>

- Standardizing apart eliminates overlap of variables, e.g., Knows($z_{17}, OJ$)
Unification

- We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

$\theta = \{x/\text{John}, y/\text{John}\}$ works

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John},x)$</td>
<td>$\text{Knows}(\text{John},\text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John},x)$</td>
<td>$\text{Knows}(y,\text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John},x)$</td>
<td>$\text{Knows}(y,\text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John},x)$</td>
<td>$\text{Knows}(x,\text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(z_{17},\text{OJ})$
Unification

- We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$

\[ \theta = \{x/\text{John}, y/\text{John}\} \text{ works} \]

- $\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td>${y/\text{John}, x/\text{Mother}(\text{John})}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
Unification

- We can get the inference immediately if we can find a substitution \( \theta \) such that \( \text{King}(x) \) and \( \text{Greedy}(x) \) match \( \text{King}(\text{John}) \) and \( \text{Greedy}(y) \)

\( \theta = \{x/\text{John}, y/\text{John}\} \) works

- \( \text{Unify}(\alpha, \beta) = \theta \) if \( \alpha\theta = \beta\theta \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(\text{John}, \text{Jane}) )</td>
<td>{x/\text{Jane}}</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{OJ}) )</td>
<td>{x/\text{OJ}, y/\text{John}}</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{Mother}(y)) )</td>
<td>{y/\text{John}, x/\text{Mother}(\text{John})}</td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(x, \text{OJ}) )</td>
<td>{fail}</td>
</tr>
</tbody>
</table>

- **Standardizing apart** eliminates overlap of variables, e.g., \( \text{Knows}(z_{17}, \text{OJ}) \)
Unification

- To unify $\text{Knows}(\text{John}, x)$ and $\text{Knows}(y, z)$,

$$\theta = \{ y/\text{John}, x/z \} \text{ or } \theta = \{ y/\text{John}, x/\text{John}, z/\text{John} \}$$

- The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.

$$\text{MGU} = \{ y/\text{John}, x/z \}$$
The unification algorithm

```
function UNIFY(x, y, θ) returns a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound
        y, a variable, constant, list, or compound
        θ, the substitution built up so far

if θ = failure then return failure
else if x = y then return θ
else if VARIABLE?(x) then return UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) then return UNIFY-VAR(y, x, θ)
else if COMPOUND?(x) and COMPOUND?(y) then
    return UNIFYARGS(x, y, UNIFY(OP[x], OP[y], θ))
else if LIST?(x) and LIST?(y) then
    return UNIFYREST(x, y, UNIFYFIRST(x, y, θ))
else return failure
```
The unification algorithm

function UNIFY-VAR(var, x, θ) returns a substitution
inputs: var, a variable
         x, any expression
         θ, the substitution built up so far

if \{var/val\} ∈ θ then return UNIFY(val, x, θ)
else if \{x/val\} ∈ θ then return UNIFY(var, val, θ)
else if OCCUR-CHECK?(var, x) then return failure
else return add \{var/x\} to θ
Automated Deduction [1]:
Sequent Rules for FOL

Sound inference: find $\alpha$ such that $KB \models \alpha$.
Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

$$
\frac{\alpha, \alpha \Rightarrow \beta \quad At(Joe, UCB) \quad At(Joe, UCB) \Rightarrow OK(Joe)}{\beta \quad OK(Joe)}
$$

E.g., And-Introduction (AI)

$$
\frac{\alpha \quad \beta \quad OK(Joe) \quad CSMajor(Joe)}{\alpha \land \beta \quad OK(Joe) \land CSMajor(Joe)}
$$

E.g., Universal Elimination (UE)

$$
\frac{\forall x \quad \alpha \quad \forall x \quad At(x, UCB) \Rightarrow OK(x)}{\alpha\{x/\tau\} \quad At(Pat, UCB) \Rightarrow OK(Pat)}
$$

$\tau$ must be a ground term (i.e., no variables)
Automated Deduction [2]: Example Proof

- Bob is a buffalo
  Pat is a pig
  Buffaloes outrun pigs
  Bob outruns Pat

1. Buffalo(Bob)
2. Pig(Pat)
3. \( \forall x, y \) Buffalo(x) \& Pig(y) \Rightarrow Faster(x, y)

Apply Sequent Rules to Generate New Assertions

Al 1 & 2

4. Buffalo(Bob) \& Pig(Pat)

UE 3, \{x/Bob, y/Pat\}

5. Buffalo(Bob) \& Pig(Pat) \Rightarrow Faster(Bob, Pat)

MP 6 & 7

6. Faster(Bob, Pat)

\[
\alpha, \quad \alpha \Rightarrow \beta \\
\hline
\beta
\]

\[
\alpha \quad \beta \\
\hline
\alpha \wedge \beta
\]

\[
\forall x \quad \alpha \\
\hline
\alpha\{x/\tau\}
\]

- Modus Ponens
- And Introduction
- Universal Elimination
**Generalized Modus Ponens (GMP)**

\[
p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)
\]

\[
\frac{}{q^\theta}
\]

where \( p_i'^\theta = p_i^\theta \) for all \( i \)

\( p_1' \) is \( \text{King}(\text{John}) \)  \( p_1 \) is \( \text{King}(x) \)
\( p_2' \) is \( \text{Greedy}(y) \)  \( p_2 \) is \( \text{Greedy}(x) \)
\( \theta \) is \( \{x/\text{John}, y/\text{John}\} \)  \( q \) is \( \text{Evil}(x) \)
\( q^\theta \) is \( \text{Evil}(\text{John}) \)

GMP used with KB of definite clauses (exactly one positive literal)
All variables assumed universally quantified
Example Knowledge Base [1]: English Statement of KB and Query

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal
Example Knowledge Base [2]: Rules and Facts

... it is a crime for an American to sell weapons to hostile nations:

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

Nono ... has some missiles, i.e., \(\exists x \text{ Owns}(\text{Nono}, x) \land \text{Missile}(x)\):

\(\text{Owns}(\text{Nono}, M_1)\) and \(\text{Missile}(M_1)\)

... all of its missiles were sold to it by Colonel West

\(\forall x \text{ Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})\)

Missiles are weapons:

\(\text{Missile}(x) \Rightarrow \text{Weapon}(x)\)

An enemy of America counts as "hostile":

\(\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)\)

West, who is American ...

\(\text{American}(\text{West})\)

The country Nono, an enemy of America ...

\(\text{Enemy}(\text{Nono}, \text{America})\)
Forward Chaining in FOL [2]: Example Proof

Diagram:
- Criminal(West)
  - Weapon(M1)
  - Sells(West,M1,Nono)
  - Hostile(Nono)
    - American(West)
    - Missile(M1)
    - Owns(Nono,M1)
    - Enemy(Nono,America)
Forward Chaining in FOL [3]: Properties

Sound and complete for first-order definite clauses (proof similar to propositional proof)

Datalog = first-order definite clauses + no functions (e.g., crime KB)
FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

May not terminate in general if $\alpha$ is not entailed

This is unavoidable: entailment with definite clauses is semidecidable
Forward Chaining in FOL [4]: Efficiency

Simple observation: no need to match a rule on iteration $k$
if a premise wasn’t added on iteration $k - 1$

$\Rightarrow$ match each rule whose premise contains a newly added literal

Matching itself can be expensive

Database indexing allows $O(1)$ retrieval of known facts
e.g., query $\text{Missile}(x)$ retrieves $\text{Missile}(M_1)$

Matching conjunctive premises against known facts is NP-hard

Forward chaining is widely used in deductive databases
Forward Chaining in FOL [1]: Algorithm

function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty
    new ← {} 
    for each sentence r in KB do 
        (p₁ ∧ ... ∧ pₙ ⇒ q) ← STANDARDIZE-A-PART(r) 
        for each θ such that (p₁ ∧ ... ∧ pₙ)θ = (p₁' ∧ ... ∧ pₙ')θ for some p₁', ..., pₙ' in KB 
            q' ← subst(θ, q) 
            if q' is not a renaming of a sentence already in KB or new then do 
                add q' to new 
                φ ← UNIFY(q', α) 
                if φ is not fail then return φ 
            add new to KB 
    return false
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
inputs: KB, a knowledge base
        goals, a list of conjuncts forming a query (θ already applied)
        θ, the current substitution, initially the empty substitution {} 
local variables: answers, a set of substitutions, initially empty
if goals is empty then return {θ}
q' ← SUBST(θ, FIRST(goals))
for each sentence r in KB
    where STANDARDIZE-APART(r) = (p₁ ∧ ... ∧ pₙ ⇒ q)
    and θ' ← UNIFY(q, q') succeeds
    new_goals ← [p₁, ..., pₙ | REST(goals)]
    answers ← FOL-BC-Ask(KB, new_goals, COMPOSE(θ', θ)) ∪ answers
return answers
Example Knowledge Base [2]: Rules and Facts

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]

Missiles are weapons:
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]

An enemy of America counts as "hostile":
\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]

West, who is American ...
\[ \text{American}(\text{West}) \]

The country Nono, an enemy of America ...
\[ \text{Enemy}(\text{Nono}, \text{America}) \]
Backward Chaining in FOL [2]: Example Proof
Depth-first recursive proof search: space is linear in size of proof

Incomplete due to infinite loops
  ⇒ fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure)
  ⇒ fix using caching of previous results (extra space!)

Widely used (without improvements!) for logic programming
Resolution: Brief Summary

Full first-order version:

\[
\begin{align*}
\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n \\
(\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta
\end{align*}
\]

where \( \text{UNIFY}(\ell_i, \neg m_j) = \theta \).

For example,

\[
\begin{align*}
\neg \text{Rich}(x) \lor \text{Unhappy}(x) \\
\text{Rich}(\text{Ken}) \\
\hline
\text{Unhappy}(\text{Ken})
\end{align*}
\]

with \( \theta = \{x/\text{Ken}\} \)

Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Conversion of FOL to Clausal Form (CNF) [1]

Everyone who loves all animals is loved by someone:
\[ \forall x \ [\forall y \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y \ Loves(y, x)] \]

1. Eliminate biconditionals and implications

\[ \forall x \ [\neg \forall y \ \neg Animal(y) \lor Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]

2. Move \(\neg\) inwards:
\[ \neg \forall x, p \equiv \exists x \ \neg p, \ \neg \exists x, p \equiv \forall x \ \neg p: \]
\[ \forall x \ [\exists y \ \neg (\neg Animal(y) \lor Loves(x, y))] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ \neg \neg Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
Conversion of FOL to Clausal Form (CNF) [2]

3. Standardize variables: each quantifier should use a different one

$$\forall x \ (\exists y \ Animal(y) \land \neg Loves(x, y)) \lor (\exists z \ Loves(z, x))$$

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$$\forall x \ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

5. Drop universal quantifiers:

$$[Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x)$$

6. Distribute $\land$ over $\lor$:

$$[Animal(F(x)) \lor Loves(G(x), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(x), x)]$$
Resolution Mnemonic: INSEUDOR

- Implications Out (Replace with Disjunctive Clauses)
- Negations Inward (DeMorgan’s Theorem)
- Standardize Variables Apart (Eliminate Duplicate Names)
- Existentials Out (Skolemize)
- Universals Made Implicit
- Distribute And Over Or (i.e., Disjunctions Inward)
- Operators Made Implicit (Convert to List of Lists of Literals)
- Rename Variables (Independent Clauses)
- A Memonic for Star Trek: The Next Generation Fans

Captain Picard:

I’ll Notify Spock’s Eminent Underground Dissidents On Romulus
I’ll Notify Sarek’s Eminent Underground Descendant On Romulus
Resolution Proof: Definite Clauses

\[ \neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x) \]

\[ \neg \text{Criminal}(\text{West}) \]

\[ \neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(x) \lor \text{Weapon}(x) \]

\[ \neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono}) \]

\[ \neg \text{Sells}(\text{West},\text{M1},x) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(\text{M1}) \lor \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Enemy}(\text{Nono},\text{America}) \]

\[ \text{Enemy}(\text{Nono},\text{America}) \]
TERMINOLOGY

- **Generalized Modus Ponens (GMP)**
  - Sound and complete rule for first-order inference (reasoning in FOL)
  - Requires pattern matching by unification

- **Unification: Algorithm for Matching Patterns**
  - Used in type inference, first-order inference
  - Matches well-formed formulas (WFFs), atoms, terms
  - Terms: variables, constants, functions and arguments
  - Arguments: nested terms

- **Resolution: Sound and Complete Inference Rule/Procedure for FOL**
  - Antecedent (aka precedent): sentences “above line” in sequent rule
  - Resolvent (aka consequent): sentences “below line” in sequent rule

- **Forward Chaining: Systematic Application of Rule to Whole KB**
  - Rete algorithm in production systems for expert systems development
  - Susceptible to high fan-out (branch factor)

- **Backward Chaining: Goal-Directed**
Ontologies & Knowledge Representation
Ontology (information science)

This article is about ontology in information science and computer science. For the term in philosophy, see ontology.

In computer science and information science, an ontology is a formal representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain.

In theory, an ontology is a "formal, explicit specification of a shared conceptualisation". An ontology provides a shared vocabulary, which can be used to model a domain — that is, the type of objects and/or concepts that exist, and their properties and relations.

Ontologies are used in artificial intelligence, the Semantic Web, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it.

© 2009 Wikipedia.

Ontologies usually are Description Logics

- e.g. Wordnet, Framenet, YAGO, Freebase
- A family of logic-based Knowledge Representation Formalisms
  - Descendents of semantic networks and KL-ONE
  - Describe domain in terms of concepts (classes), Roles (properties, relationships) and individuals
- Distinguished by
  - Formal Semantics (typically model theoretic)
  - Decidable fragments of FOL
  - Provision of inference services, e.g. Decision procedures for key problems (satisfiability, subsumption)
  - Implemented Systems (highly optimized)
Description Logics Basics

• Concepts (formulae)
  • e.g. person, doctor, happyparent,
• Roles e.g. haschild, loves
• Individuals (nominals): John, Mary, Italy
• Operators (for forming concepts and roles) restricted so that
  • Satisfiability is decidable and if possible of low complexity
  • No need for explicit use of variables, i.e., restrictions on existential and universal quantification
What Is A “Concept”? 

- “Concept” and “Class” are used synonymously
- A class is a **concept** in the domain
  - a class of wines
  - a class of wineries
  - a class of red wines
- A class is a **collection** of elements with similar properties
- **Instances** of classes
  - a glass of California wine you’ll have for lunch
Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)

A class hierarchy is usually an IS-A hierarchy:

- An instance of a subclass is an instance of a superclass

- If you think of a class as a set of elements, a subclass is a subset
**Class Inheritance — Example**

- Apple is a subclass of Fruit
  - *Every apple is a fruit*
- Red wine is a subclass of Wine
  - *Every red wine is a wine*
- Chianti wine is a subclass of Red wine
  - *Every Chianti wine is a red wine*
Levels In The Hierarchy

Top level:
- Wine
  - White wine
  - Rosé wine
  - Red wine
    - Beaujolais
    - Red Burgundy
    - Red Zinfandel
  - Red Bordeaux
    - Medoc
      - Pauillac
      - Margaux
    - St. Emillion
    - Graves
    - Cabernet Franc
    - Cabernet Sauvignon
  - Pinot Noir
  - Chianti
  - Petite Syrah
  - Sancerre
  - Muscadet

Middle level:

Bottom level:
Defining Properties of Classes: Slots

- Slots, Attributes, and Relations: synonymous
- Slots in class definition $C$
  - Describe attributes of instances of $C$
  - Describe relationships to other instances

```
Slots for the Concept/Class Wine
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Other Facets</th>
</tr>
</thead>
<tbody>
<tr>
<td>body</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={FULL, MEDIUM, LIGHT}</td>
</tr>
<tr>
<td>color</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={RED, ROSÉ, WHITE}</td>
</tr>
<tr>
<td>flavor</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DELICATE, MODERATE, STRONG}</td>
</tr>
<tr>
<td>grape</td>
<td>Instance</td>
<td>multiple</td>
<td>classes={Wine grape}</td>
</tr>
<tr>
<td>maker</td>
<td>Instance</td>
<td>single</td>
<td>classes={Winery}</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td>single</td>
<td></td>
</tr>
<tr>
<td>sugar</td>
<td>Symbol</td>
<td>single</td>
<td>allowed-values={DRY, SWEET, OFF-DRY}</td>
</tr>
</tbody>
</table>
Types of properties

- “intrinsic” properties: flavor and color of wine
- “extrinsic” properties: name and price of wine
- parts: ingredients in a dish
- relations to other objects: producer of wine (winery)

Simple and complex properties

- simple properties (attributes): contain primitive values (strings, numbers)
- complex properties: contain (or point to) other objects (e.g., a winery instance)
A subclass inherits all the slots from the superclass
- If a wine has a name and flavor, a red wine also has a name and flavor

If a class has multiple superclasses, it inherits slots from all of them
- Port is both a dessert wine and a red wine. It inherits “sugar content: high” from the former and “color:red” from the latter
Property constraints (facets) describe or limit the set of possible values for a slot

- The name of a wine is a string
- The wine producer is an instance of Winery
- A winery has exactly one location

Facets for slots in the **Wine** class
What is required of a knowledge representation?

- Representational adequacy: It should allow you to represent all the knowledge you need to reason with.
- Inferential adequacy: It should allow new knowledge to be inferred from a basic set of facts.
- Inferential efficiency: Inferences should be made efficiently.
- Clear Syntax and Semantics: We should know what the allowable expressions of the language are and what they mean.
- Naturalness: The language should be reasonably natural and easy to use.
Building a knowledge base

- Knowledge engineering (Circa 1980’s)
  - Investigate domain
  - Create formal representation of objects
  - Interview domain experts
    - Knowledge Acquisition & Representation (TAKES YEARS)
- [http://wordnetweb.princeton.edu/perl/webwn](http://wordnetweb.princeton.edu/perl/webwn)

Now: Crowd Source It! Data Mine it! Both!
WordNet (already part of NLTK)

- 70,000 synsets (synonym sets)
  - Simple, noun hierarchy

- Widely used in language processing:
  - Query expansion, IR, Translation
  - Online version
    http://www.cogsci.princeton.edu/cgi-bin/webwn
WordNet 2.0 Search

Search word: greenhouse Find senses

Overview for "greenhouse"

The noun "greenhouse" has 1 sense in WordNet.

1. greenhouse, nursery, glasshouse -- (a building with glass walls and roof; for the cultivation and exhibition of plants under controlled conditions)

Search for Synonyms, ordered by estimated frequency of senses

☑ Show glosses
☐ Show contextual help
Search

The adjective "greenhouse" has 1 sense in WordNet.

1. greenhouse -- (of or relating to or caused by the greenhouse effect; "greenhouse gases")

Search for Synonyms/Related Nouns of senses

☑ Show glosses
☐ Show contextual help
Search
WordNet Resources

- “greenhouse” as noun (X)
  - Synonyms, “nursery”
  - Hypernyms (designating whole class, X IS_A), e.g., “building”
  - Hyponyms (designating subclass, Y IS_KIND_OF), e.g., “orangery”, “conservatory”
  - Meronyms (constituent parts or substance, X HAS_PART), e.g., “door”, “wall”
  - Polysemy count (number of senses of word in a syntactic category)
Verb *get* has 36 senses

1. **get, acquire** - come into the possession of something concrete or abstract; "She got a lot of paintings from her uncle"; "They acquired a new pet"; "Get your results the next day"; "Get permission to take a few days off from work"
   - Derived form: noun *getting*
   - Sample sentence:
     The children get the ball.
2. **become, go, get** - enter or assume a certain state or condition; "He became annoyed when he heard the bad news"; "It must be getting more serious"; "her face went red with anger"; "She went into ecstasy"; "Get going!"
   - 2 is one way to change state, turn
   - Sample sentence:
     John will get angry.
3. **get, let, have** - cause to move; cause to be in a certain position or condition; "He got his squad on the ball"; "This let me in for a big surprise"; "He got a girl into trouble"
   - 3 is one way to make, get
   - Sample sentences:
     Somebody ----s somebody PP
     Somebody ----s something PP
     Somebody ----s somebody to INFINITIVE
4. **receive, get, find, obtain, incur** - receive a specified treatment (abstract); "These aspects of civilization do not find expression or receive an interpretation"; "His movie received a good review"; "I got nothing but trouble for my good intentions"
   - 4 is one way to change
   - Sample sentence:
     Something ----s
5. **arrive, get, come** - reach a destination; arrive by movement or progress; "She arrived home at 7 o'clock"; "She didn't get to Chicago until after midnight"
   - Sample sentence:
Add to it! (Snow, Jurafsky et al)

Example: Using the “Y called X” Pattern for Hypernym Acquisition


None of the following links are contained in WordNet (or the training set).

<table>
<thead>
<tr>
<th>Hyponym</th>
<th>Hypernym</th>
<th>Sentence Fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>efflorescence</td>
<td>condition</td>
<td>...and a condition called efflorescence…</td>
</tr>
<tr>
<td>’neal_inc</td>
<td>company</td>
<td>...The company, now called O’Neal Inc…</td>
</tr>
<tr>
<td>hat_creek_outfit</td>
<td>ranch</td>
<td>...run a small ranch called the Hat Creek Outfit.</td>
</tr>
<tr>
<td>tardive_dyskinesia</td>
<td>problem</td>
<td>...irreversible problem called tardive dyskinesia…</td>
</tr>
<tr>
<td>hiv-1</td>
<td>aids_virus</td>
<td>...infected by the AIDS virus, called HIV-1.</td>
</tr>
<tr>
<td>bateau_mouche</td>
<td>attraction</td>
<td>...sightseeing attraction called the Bateau Mouche…</td>
</tr>
<tr>
<td>kibbutz_malkiyya</td>
<td>collective_farm</td>
<td>...Israeli collective farm called Kibbutz Malkiyya…</td>
</tr>
</tbody>
</table>
All you ever wanted to know about YAGO
But we're afraid to ask...

By Ben Samuel
Inspired by the paper by Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum
YAGO—what?

- YAGO = Yet Another Great Ontology.
- Great! What does Ontology mean?
- In this case, it means a collection of Facts
  - Such as…
    - Elvis’ Birthday
    - What London is called in French
    - And many more!

Sounds Like Fun! But is it useful?
Yes! Ontologies are Wonderful!

- Ontologies are used all over the place, helping to fuel a variety of helpful tools.
  - Word Sense Disambiguation
    - Kinda like Scribblenauts!
  - Question Answering and information retrieval
    - Wine and Nutrition Chatbots we read about.
  - Machine Translation
    - Francais to French!
  - Document Classification
    - Categorize Electronic Documents!

Thank you Ontologies!
So where does YAGO get the goods?

- Lots of existing ontologies get their facts from a single source
  - Frequently domain dependent – Limiting!
  - Not as high quality or well structured as multiple sources would be.

- YAGO takes information from both Wikipedia and WordNet
  - Wikipedia provides vast amounts of data on actual people
  - WordNet provides a Taxonomy, which enables internal classification of said data.
An Old Friend...

YAGO (ontology)
From Wikipedia, the free encyclopedia
(Redirected from YAGO (Ontology))

YAGO (Yet Another Great Ontology) is a huge semantic knowledge base. Currently, YAGO knows over two million entities such as persons, organizations and cities and about twenty million facts about these entities. A web interface allows users to pose questions to YAGO in the form of queries on the YAGO homepage. YAGO is being developed at the Max Planck Institute for Computer Science.[1]

YAGO is automatically extracted from Wikipedia and uses WordNet to structure information.

See also

- DBpedia
- Semantic Web

References


External links

[edit]
And a New Ally?


Word to search for: Pizza
Display Options: (Select option to change) · Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- **S:** (n) pizza, pizza pie (Italian open pie made of thin bread dough spread with a spiced mixture of e.g. tomato sauce and cheese)
  - *direct hyponym / full hyponym*
    - **S:** (n) sausage pizza (tomato and cheese pizza with sausage)
    - **S:** (n) pepperoni pizza (tomato and cheese pizza with pepperoni)
    - **S:** (n) cheese pizza (pizza with lots of cheese)
    - **S:** (n) anchovy pizza (tomato and cheese pizza with anchovies)
    - **S:** (n) Sicilian pizza (pizza made with a thick crust)
  - *direct hypernym / inherited hypernym / sister term*

WordNet home page
Fun with Facts

- So, just how are facts represented inside of YAGO?
  - Every “object” (Elvis, Einstein, 1984, France, Pizza) is known as an *Entity*
  - Two Entities can stand in a *Relation* to each other.

  - For example:
    - ElvisPresley HasWonPrize Grammy Award
    - singer SubClassOf person
    - “Elvis” means Elvis Presley
More Facts about Facts

- YAGO also gives each fact a unique identifier. If (ElvisPresley, HasWonPrize, Grammy Award) has fact identifier #1 then...

- (#1, FoundIn, Wikipedia) means that the fact was found in Wikipedia.

- Becomes very important for representing n-ary relations. For example:

- #2: #1 InYear 1967 Would Mean: “Fact number 2 in the ontology is that Elvis Presley Has won a Grammy Award in 1967”
Of course, with all of these great facts living in YAGO, we need a way to get them out!

Good thing we have a *Query Language*

- So, to find out what year Elvis won the Grammy, for example...

  ?i1: Elvis HasWonPrize Grammy Award
  ?i2: ?i1 InYear ?x

  The year 1967 should get bound to ?x
And HOW does YAGO get the goods?

- **Wikipedia: Focuses on InfoBoxes**
  - Structured Data is easier to gather, doesn’t have to deal with natural language understanding.
  - Still needs to take context of page into account though, by looking at ‘type’ of infobox.
    - The *length* of a car is different than the *length* of a song, for example. (Car infobox vs. song infobox)

- **WordNet: Looks at Synsets**
  - Each Synset becomes a class in YAGO (e.g. person)
Two Great Tastes that Go Great Together

- Connects the two by using the hyponyms/hypernyms of WordNet
  - Hyponym: X is a type of Y.
  - Hypernym: A type of X is Y.

- A class is a subset of another if the first is a hyponym of the second.
  - singer SubClassOf person

- Algorithmically connect “low” Wikipedia classes (American people in Japan) to “high” WordNet classes (Person)
YAGO was tested by giving human judges randomly selected facts, and asking them if YAGO had it right
  ◦ Were provided with Wikipedia Context

Evaluation had terrific results of around 95% accuracy!
  ◦ The crucial link between WordNet and Wikipedia turned out to be very accurate!
    • Some “errors” were more philosophical in nature (What does it mean to be a *French Economist*)
‘Leggo my YAGO!

- YAGO is very large, and growing all the time!
  - 1.7M Entities, 15M Facts in 2008

- And people are using it! (either by borrowing entities, leveraging it’s taxonomic information, or including YAGO all together!)
  - Various research projects
    - Semantic Search, Entity Organization, Information Extraction
  - Various Ontology projects
    - Freebase, UMBEL, Cyc, Dbpedia.
In Short:

- YAGO is an ontology which consists of many, many facts.
- It forms the facts by leveraging information from Wikipedia and WordNet.
- Has a query language to retrieve facts.
- Is used by many other projects, and seems to be growing rapidly itself. (online demo: http://www.mpii.de/yago)
Next Time: Classical Planning
Read Chapter 10 before class.
The DPLL algorithm

EVERY FOL KB can be converted to a PL KB

Determine if an input propositional logic sentence (in CNF) is satisfiable.

Improvements over truth table enumeration:

1. **Early termination**
   A clause is true if any literal is true.
   A sentence is false if any clause is false.

2. **Pure symbol heuristic**
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses (A ∨ ¬B), (¬B ∨ ¬C), (C ∨ A), A and B are pure, C is impure.
   Make a pure symbol literal true.

3. **Unit clause heuristic**
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.