Knowledge Representation

CS 486/686: Introduction to Artificial Intelligence
Fall 2013
Outline

• Knowledge-based agents
• Logics in general
• Propositional Logic
• Reasoning with Propositional Logic
• First Order Logic
Introduction

- So far we have taken the following approach
  - Figure out exactly what the problem is (problem definition)
  - Design an algorithm to solve the problem (search algorithm)
  - Execute the program
Knowledge-Based Agents

- An alternative approach
  - Identify the knowledge needed to solve the problem
  - Write down this knowledge in some language
  - Use logical consequences to solve the problem
Knowledge-Based Agents

• Ideally
  - We tell the agent what it needs to know
  - The agent infers what to do and how to do it

• Agent has two parts
  - **Knowledge base**: Set of facts expressed in a formal standard language
  - **Inference engine**: Rules for deducing new facts
An Example: Wumpus World

• **Goal:**
  – Get gold back to start without falling into a pit or getting eaten by the wumpus

• **Environment**
  – Squares adjacent to wumpus are smelly
  – Squares adjacent to pit are breezy
  – Glitter iff gold is in the same square
  – Shooting kills wumpus if you are facing it
  – Shooting uses up the only arrow
  – Grabbing picks up gold if in same square
  – Releasing drops the gold in same square

• **Sensors:** Stench, Breeze, Glitter, Bump, Scream

• **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot
Wumpus World

<table>
<thead>
<tr>
<th>1,1</th>
<th>2,1</th>
<th>3,1</th>
<th>4,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,2</th>
<th>2,2</th>
<th>3,2</th>
<th>4,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,3</th>
<th>2,3</th>
<th>3,3</th>
<th>4,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>1,1</th>
<th>2,1</th>
<th>3,1</th>
<th>4,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>B</td>
<td>P!</td>
<td>4,1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,2</th>
<th>2,2</th>
<th>3,2</th>
<th>4,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,3</th>
<th>2,3</th>
<th>3,3</th>
<th>4,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>W!</td>
<td>W!</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>1,1</th>
<th>2,1</th>
<th>3,1</th>
<th>4,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,2</th>
<th>2,2</th>
<th>3,2</th>
<th>4,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>V</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,3</th>
<th>2,3</th>
<th>3,3</th>
<th>4,3</th>
</tr>
</thead>
<tbody>
<tr>
<td>W!</td>
<td>W!</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1,4</th>
<th>2,4</th>
<th>3,4</th>
<th>4,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
</tbody>
</table>

---

**Legend:**
- **Agent**
- **Breeze**
- **Glitter, Gold**
- **Safe square**
- **Pit**
- **Stench**
- **Visited**
- **Wumpus**
What Is A Logic?

• Logic
  - A formal language for representing information so that conclusions can be drawn

• Logics have 2 components
  - Syntax: defines the sentences of the language
  - Semantics: defines the meaning of the sentences
Entailment

- Entailment means that "one thing follows from another"
  - KB |= α

- Knowledge base (KB) entails sentence α if and only if α is true in all possible worlds where KB is true

- Example:
  - KB: I finished the AI assignment. I am happy
  - α: I finished the AI assignment and I am happy.
Models

• A model is a formal “possible world” where a sentence can be evaluated
  - \( m \) is a model of sentence \( \alpha \) if \( \alpha \) is true in \( m \)

• \( M(\alpha) \) is the set of all models of \( \alpha \)

• \( KB \models \alpha \) if and only if \( M(KB) \subseteq M(\alpha) \)

KB: I finished the AI homework and I did not sleep last night
\( \alpha \): I finished the AI homework
Inference

- Given a KB, we want to be able to draw conclusions from it

- **Inference procedure:** KB |-i α
  - Sentence α can be derived from KB by inference algorithm

- Desired properties:
  - **Soundness:** the procedure only infers true statements
    - If KB |-i α then KB |= α
  - **Completeness:** the procedure can generate all true statements
    - IF KB |= α then it is true that KB|-i α
Propositional Logic

- **Atomic Symbols**: P, Q, R,...
  - Each symbol stands for a proposition that can be either True or False

- **Logical Connectives**
  - \( \neg \) (negation)
  - \( \lor \) (or)
  - \( \land \) (and)
  - \( \Rightarrow \) (implies)
  - \( \Leftrightarrow \) (if and only if, equivalence)
Propositional Logic: Syntax

• Grammar rules:
  – Sentence → AtomicSentence | ComplexSentence
  – Atomic Sentence → True | False | Symbol
  – Symbol → P | Q | R | ...
  – ComplexSentence → Sentence I ¬ Sentence
  I (Sentence ∨ Sentence)
  I(Sentence ∧ Sentence)
  I (Sentence ⇒ Sentence)
  I(Sentence ⇔ Sentence)
Propositional Logic: Semantics

• The semantics of propositional logic are defined by a truth table
  - Symbols are mappings to an element in domain \{0,1\}

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P⇔Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example: Propositional Logic

• Note that $P \Rightarrow Q$ is the same as $\neg P \lor Q$

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>$\neg P$</th>
<th>$\neg P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Exercise: Show that $P \Leftrightarrow Q$ is the same as $(P \Rightarrow Q) \land (Q \Rightarrow P)$
Entailment: Propositional Logic

- Let
  - $KB = (P \lor R) \land (Q \lor \neg R)$
  - $\alpha = P \lor R$

- Does $KB \models \alpha$?
Entailment

- Check all possible models
  - \( \alpha \) must be true whenever \( KB \) is true

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>( P \lor R )</th>
<th>( Q \lor \neg R )</th>
<th>KB</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Inference: Propositional Logic

- Using truth tables is
  - **Sound**: direct definition of entailment
  - **Complete**: works for any KB and α and always terminates

- But...
  - Really inefficient
  - If there are n symbols, then there are $2^n$ models
More Terminology

• Sentences $\alpha$ and $\beta$ are logically equivalent if they are true in the same set of models
  - $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

• Deduction Theorem:
  - For any sentences $\alpha$ and $\beta$, $\alpha \models \beta$ if and only if $(\alpha \Rightarrow \beta)$ is valid

• Useful Result (Proof by Contradiction):
  - $\alpha \models \beta$ if and only if the sentence $(\alpha \land \neg \beta)$ is unsatisfiable
Logical Equivalences

\[(\alpha \land \beta) \equiv (\beta \land \alpha)\]  \text{commutativity of } \land

\[(\alpha \lor \beta) \equiv (\beta \lor \alpha)\]  \text{commutativity of } \lor

\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma))\]  \text{associativity of } \land

\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma))\]  \text{associativity of } \lor

\[\lnot(\lnot \alpha) \equiv \alpha\]  \text{double-negation elimination}

\[(\alpha \Rightarrow \beta) \equiv (\lnot \beta \Rightarrow \lnot \alpha)\]  \text{contraposition}

\[(\alpha \Rightarrow \beta) \equiv (\lnot \alpha \lor \beta)\]  \text{implication elimination}

\[(\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))\]  \text{biconditional elimination}

\[\lnot(\alpha \land \beta) \equiv (\lnot \alpha \lor \lnot \beta)\]  \text{de Morgan}

\[\lnot(\alpha \lor \beta) \equiv (\lnot \alpha \land \lnot \beta)\]  \text{de Morgan}

\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma))\]  \text{distributivity of } \land \text{ over } \lor

\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))\]  \text{distributivity of } \lor \text{ over } \land
Inference Rules

• Given a KB we want to derive conclusions
  - Proof: sequence of inference rule applications

\[
\begin{align*}
\text{Modus Ponens} & : \\
\alpha, \alpha \Rightarrow \beta & : \\
& \quad \beta
\end{align*}
\]

\[
\begin{align*}
\text{Resolution} & : \\
\alpha \lor \beta, \neg \beta \lor \gamma & : \\
& \quad \alpha \lor \gamma
\end{align*}
\]

\[
\begin{align*}
\text{And Elimination} & : \\
\alpha \land \beta & :\\
& \quad \alpha
\end{align*}
\]

\[
\begin{align*}
\text{Unit Resolution} & : \\
\alpha \lor \beta, \neg \beta & : \\
& \quad \alpha
\end{align*}
\]
Inference

- Modus Ponens and And-Elimination together are sound

**Example:** \( \text{KB}= \text{“If you are in AI class then you are happy and paying attention”, “You are in AI class”} \)

**Modus Ponens:** “You are happy and you are paying attention”

**And-Elimination:** “You are happy”
Resolution

- Resolution is a **sound** and **complete** inference rule
  
  - Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.
Resolution

- Resolution is a **sound and complete** inference rule
  - Any complete search algorithm, applying only the resolution rule, can derive any conclusion entailed by any knowledge base in propositional logic.

**Caveat:** Given that $\alpha$ is true, we cannot automatically generate $\alpha \lor \beta$ is true. However, we can find the answer to the question “Is $\alpha \lor \beta$ true?”
Conjunctive Normal Form

- Resolution is applied to clauses of the form $\alpha \lor \beta \lor \ldots \lor \gamma$

- Any clause in propositional logic is logically equivalent to a clause in CNF
  - conjunction of disjunctions
  - eg. $(P \lor \neg Q \lor R) \land (\neg Q \lor A \lor B) \land \ldots$
Converting to CNF

1. Eliminate $\iff$, replacing $P \iff Q$ with

   $$(P \implies Q) \land (Q \implies P)$$

2. Eliminate $\implies$, replacing $P \implies Q$ with $\neg P \lor Q$

3. Move "\neg" inwards, using $\neg(\neg P) = P$,
   $\neg(P \land Q) = \neg P \lor \neg Q$ and $\neg(P \lor Q) = \neg P \land \neg Q$

4. Distribute $\lor$ over $\land$ where possible
Resolution Algorithm

• Recall: To show $\text{KBI} = \alpha$, we show that $(\text{KB} \land \lnot \alpha)$ is unsatisfiable

• Resolution Algorithm:
  - Convert $(\text{KB} \land \lnot \alpha)$ to CNF
  - For every pair of clauses that contain complementary literals
    - Apply resolution to produce a new clause
    - Add new clause to set of clauses
  - Continue until
    • No new clauses are being added (KB does not entail $\alpha$) or
    • Two clauses resolve to produce empty clause (KBI=$\alpha$)
Complexity of Inference

- Inference for propositional logic is NP-complete
- If all clauses are **Horn clauses**, then inference is linear in size of KB!
  - Horn clause: Disjunction of literals where at most one literal is positive
    - \( \neg P \lor Q \lor \neg R \) is a Horn clause
    - \( P \lor Q \lor R \) is not a Horn clause
  - Every Horn clause establishes exactly one new fact
    - \( \neg P \lor Q \lor \neg R \Leftrightarrow (P \land R) \Rightarrow Q \)
    - We add all new facts in n passes
Forward Chaining

• When a new sentence $\alpha$ is added to the KB
  - Look for all sentences that share literals with $\alpha$
  - Perform resolution
  - Add new sentence to KB and continue

• Forward chaining is
  - Data-driven
  - Eager: new facts are inferred as soon as possible
Backward Chaining

• When a query q is asked of the KB
  - If q is in the KB, return True
  - Otherwise, use resolution for q with other sentences in the KB and continue from result

• Backward chaining is
  - Goal driven: Centers reasoning around query being asked
  - Lazy: new facts are inferred only when needed
Forward vs Backward

• Which is better? That depends!

• Backward Chaining:
  - Does not grow the KB as much
  - Focused on proof so is generally more efficient
  - Does nothing until a question is asked
  - Typically used in proofs by contradiction
Forward vs Backward

- Forward Chaining
  - Extends the KB and improves understanding of the world
  - Typically used in tasks where the focus is on providing a model of the world
First Order Logic

• New elements
  - Predicates
    - Define objects, properties, relationships
  - Quantifiers
    - ∀ (for all), ∃ (there exists) are used in statements that apply to a class of objects

• Example: ∀x On(x, Table) ⇒ Fruit(x)
Sentences

• Terms
  - Constants, variables, function(term₁,...,termₙ)

• Atomic Sentences
  - Predicate(term₁,term₂), term₁=term₂

• Complex Sentences
  - Combine atomic sentences with connectives
  - Likes(Alice, IceCream) ∧ Likes(Bob, IceCream)
• Sentences are true with respect to their interpretation
  - Model contains objects and relations among them
  - Interpretation specifies referents for
    - Constant symbols (objects)
    - Predicate symbols (relations)
    - Function symbols (functional relations)
• Atomic sentence $\text{Predicate}(\text{term}_1,\ldots,\text{term}_n)$ is true if and only if the relation referred to by $\text{Predicate}$ holds for objects $\text{term}_1,\ldots,\text{term}_n$
Semantics
Universal Quantification

• Form: ∀<variables> <sentence>
  - Everyone taking AI is smart
  - ∀x Taking(x, AI) ⇒ Smart(x)

• Equivalent to
  - (Taking(Alice, AI) ⇒ Smart(Alice)) ∧ (Taking(Bob, AI) ⇒ Smart(Bob))...

• Example: ∀x Taking(x, AI) ∧ Smart(x)
  - Why is this unlikely not what you mean?

• Typically ⇒ is the main connector!
Existential Quantification

- Form: $\exists x \langle \text{variables} \rangle \langle \text{sentence} \rangle$
  - Someone is taking AI is smart
  - $\exists x \text{Taking}(x, \text{AI}) \land \text{Smart}(x)$

- Equivalent to
  - $(\text{Taking}(\text{Alice}, \text{AI}) \land \text{Smart}(\text{Alice})) \lor (\text{Taking}(\text{Bob}, \text{AI}) \land \text{Smart}(\text{Bob})) \lor \ldots$

- Example: $\exists x \text{Taking}(x, \text{AI}) \Rightarrow \text{Smart}(x)$
  - Why is this unlikely to be what you want?

- Typically $\land$ is the main connector
Properties of Quantifiers

• Basic Rules
  - \( \forall x \forall y \) is the same as \( \forall y \forall x \)
  - \( \exists x \exists y \) is the same as \( \exists y \exists x \)
  - \( \forall x \exists y \) is not the same as \( \exists y \forall x \)

• Example
  - \( \exists x \forall y \) Loves (x,y)
  - \( \forall y \exists x \) Loves(x,y)
Quantifier Duality

- Each quantifier can be expressed using the other quantifier and negation
  - $\forall x \text{ Likes}(x, \text{Broccoli})$
  - $\neg \exists x \neg \text{Likes}(x, \text{Broccoli})$
  - $\exists x \text{ Likes}(x, \text{Broccoli})$
  - $\neg \exists x \neg \text{Likes}(x, \text{Broccoli})$
Inference and FOL

- We know how to do inference in Propositional Logic: find \( \alpha \) such that \( KB \models \alpha \).
  - Is it possible to use these techniques for FOL?
  - Have to handle quantifiers, predicates, functions, ...
Universal Instantiation

- Given sentence $\forall x \ P(x) \land Q(x) \Rightarrow R(x)$ then we want to infer $P(\text{John}) \land Q(\text{John}) \Rightarrow R(\text{John})$ and $P(\text{Anne}) \land P(\text{Anne}) \Rightarrow R(\text{Anne})$ and ...

Universal Instantiation (UI)

$\forall v \alpha$  

$\alpha$ is a sentence

$\forall u \alpha$  

$\alpha$ is a sentence

$\text{SUBST} \left( \{v/g\} \alpha \right)$

Substitute $g$ for all occurrences of $v$ in $\alpha$

$v$ is a variable

$g$ is a ground term*
Universal Instantiation

• A ground term is a term without variables

• \text{SUBST}(\theta, \alpha) is the result of applying substitution \theta to sentence \alpha

• Example

  – \text{SUBST}\{{{x/John}}, \forall x P(x) \land Q(x) \Rightarrow R(x)} = P(John) \land Q(John) \Rightarrow R(John)
  
  – \text{SUBST}\{{{x/Father(John)}}, \forall x P(x) \land Q(x) \Rightarrow R(x)}
    = P(Father(John)) \land Q(Father(John)) \Rightarrow R(Father(John))
Existential Instantiation

• For any sentence $\alpha$, variable $v$ and constant symbol $K$ that does not appear anywhere in the KB

\[
\exists v \alpha \\
\text{SUBST}\left(\{x/K\}, \alpha\right)
\]

Example

$\exists x \text{Crown}(x)$ yields

$\text{Crown}(C_1)$ (\(C_1\) is a new constant)
Reduction to Propositional Inference

• Suppose the KB contained the following
  - $\forall x \, \text{Cat}(x) \land \text{Orange}(x) \Rightarrow \text{Cute}(x)$
  - $\text{Orange}(\text{Kitty})$
  - $\text{Cat}(\text{Kitty})$
  - $\text{Sister}(\text{Kitty}, \text{Katy})$

• Instantiating the universal sentence in all possible ways we have a new KB:
  - $\text{Cat}(\text{Kitty}) \land \text{Orange}(\text{Kitty}) \Rightarrow \text{Cute}(\text{Kitty})$
  - $\text{Cat}(\text{Katy}) \land \text{Orange}(\text{Katy}) \Rightarrow \text{Cute}(\text{Katy})$
  - $\text{Cat}(\text{Kitty})$
  - $\text{Sister}(\text{Kitty}, \text{Katy})$

• The new KB is in propositional form. The symbols are
  - $\text{Cat}(\text{Kitty}), \text{Cat}(\text{Katy}), \text{Orange}(\text{Kitty}), \text{Cute}(\text{Katy}), \text{Sister}(\text{Kitty}, \text{Katy}), \ldots$
Example

- KB: Bob is a buffalo. Pat is a pig. Buffalos are faster than pigs.
• Every FOL KB can be propositionalized
  - Transformed into propositional logic
• This preserves entailment
  - A ground sentence is entailed by the new KB if and only if it was entailed in the original KB
• Thus we can apply resolution (sound and complete) and return the result?
Reduction Continued

- Problem: Functions
  - The set of possible ground substitutions can be infinite
  - Example: Assume the KB contains function \( \text{Mother}(x) \)
    - \( \text{SUBST}\{\{x|\text{John}\},\text{Mother}(x)\}=\text{Mother}(\text{John}) \)
    - \( \text{SUBST}\{\{x|\text{Mother}(\text{John})\},\text{Mother}(x)\}=\text{Mother}(\text{Mother}(\text{John})) \)
    - \( \text{SUBST}\{\{x|\text{Mother}(\text{John})\},\text{Mother}(\text{Mother}(x))\}=\text{Mother}(\text{Mother}(\text{Mother}(\text{Mother}(\text{John})))) \)
• **Theorem** (Herbrand 1930): If a sentence is entailed by a FOL KB, then it is entailed by a finite subset of the propositionalized KB.

• **Idea**: for $n=0$ to $\infty$
  - Create a propositional KB by instantiating with depth $n$ terms
  - Check if $\alpha$ is entailed by this KB. If yes, then stop.
• **Problem**: Works if \( \alpha \) is entailed by the KB but it loops forever if \( \alpha \) is not entailed

• **Theorem**: (Turing 1936, Church 1936) Entailment in FOL is semi-decidable.
  - Algorithms exist that say yes to every entailed sentence
  - No algorithm exists that says no to every unentailed sentence
Problems with Propositionalization

- Problem is with universal instantiation
  - Generates many irrelevant sentences due to substitutions

- **Idea**: Find a substitution that makes different logical statements look identical
  - Unification
Unification

- **Unify algorithm**
  - Takes two sentences and returns a unifier if one exists
  - \( \text{Unify}(p,q) = \theta \) where \( \text{SUBST}(\theta,p) = \text{SUBST}(\theta,q) \)
    - \( \theta \) is the Unifier

<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,Paul)</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,Paul)</td>
<td></td>
</tr>
</tbody>
</table>
Generalized Modus Ponens

- **Conditions**: Atomic sentences $p_i$, $p_i'$ and $q$ where there is a substitution $\theta$ such that $\text{SUBST}(\theta, p_i) = \text{SUBST}(\theta, p_i')$

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

\[\text{SUBST}(\theta, q)\]

**Caveats:**

*GMP used with a KB of definite clauses* (exactly one positive literal).

All variables are assumed to be universally quantified.
Inference Algorithms

• You can now use
  - Forward chaining
  - Backward chaining
  - Resolution
Forward Chaining Example
Backward Chaining Example
Resolution Review

- Resolution is a refutation procedure
  - To prove $KB \models \alpha$ show that $KB \land \neg \alpha$ is unsatisfiable
- Resolution used $KB$, $\neg \alpha$ in CNF
- Resolution inference rule combines two clauses to make a new one

Inference continues until an empty clause is derived (contradiction)
Resolution

Where \( \text{Unify}(l_i, m_i) = \theta \)

The two clauses, \( l_i \) and \( m_i \), are assumed to be standardized apart so that they share no variables

Example

\[
\neg \text{Rich}(x) \lor \text{Unhappy}(x) \\
\text{Rich}(\text{John}) \\
\hline
\text{Unhappy}(\text{John}) \text{ with } \theta = \{x/\text{John}\}
\]
Converting to CNF

- Example $\forall x [\forall y A(y) \Rightarrow L(x,y)] \Rightarrow [\exists y L(y,x)]$
- Eliminate $\iff$ and $\Rightarrow$
  - $\forall x [\neg \forall y \neg A(y) \lor L(x,y)] \lor [\exists y L(y,x)]$
- Move $\neg$ inwards
  - $\forall x [\exists y A(y) \land \neg L(x,y)] \lor [\exists y L(y,x)]$
- Standardize variables
  - $\forall x [\exists y A(y) \land \neg L(x,y)] \lor [\exists z L(z,x)]$
- Skolemize
  - $\forall x [A(F(x)) \land \neg L(x,F(x))] \lor [L(G(x),x)]$
- Drop universal quantifiers
  - $[A(F(x)) \land \neg L(x,F(x))] \lor [L(G(x),x)]$
- Distribute $\lor$ over $\land$
  - $[A(F(x)) \lor L(G(x),x)] \land [\neg L(x,F(x)) \lor L(G(x),x)]$
Resolution Example

- Marcus is a person
- Marcus is a Pompeian
- All Pompeians are Roman
- Caesar is a ruler
- All Romans are either loyal to Caesar or hate Caesar
- Everyone is loyal to someone
- People only try to assassinate rulers they are not loyal to
- Marcus tries to assassinate Caesar
- Query: Does Marcus hate Caesar?
Conclusion

• Syntax, semantics, entailment and inference
• Propositional logic and FOL
• Understand how forward-chaining, backward-chaining and resolution work