Example

Limitations?
Rejection sampling

\( \hat{P}(X|e) \) estimated from samples agreeing with \( e \)

```plaintext
function Rejection-Sampling(X, e, bn, N) returns an estimate of \( P(X|e) \)
    local variables: N, a vector of counts over X, initially zero
    for \( j = 1 \) to \( N \) do
        x ← Prior-Sample(bn)
        if x is consistent with \( e \) then
            \( N[x] \leftarrow N[x] + 1 \) where \( x \) is the value of \( X \) in \( x \)
    return Normalize(N[X])
```

E.g., estimate \( P(Rain|Sprinkler = true) \) using 100 samples
27 samples have \( Sprinkler = true \)
Of these, 8 have \( Rain = true \) and 19 have \( Rain = false \).

\( \hat{P}(Rain|Sprinkler = true) = Normalize((8, 19)) = (0.296, 0.704) \)

Similar to a basic real-world empirical estimation procedure
Rejection Sampling as Integration

• Classic polygon problem
• In belief network, what are we integrating?
Analysis of rejection sampling

\[ \hat{P}(X|e) = \alpha N_{PS}(X, e) \quad \text{(algorithm defn.)} \]
\[ = N_{PS}(X, e)/N_{PS}(e) \quad \text{(normalized by } N_{PS}(e)) \]
\[ \approx P(X, e)/P(e) \quad \text{(property of } \text{PRIORSAMPLE}) \]
\[ = P(X|e) \quad \text{(defn. of conditional probability)} \]

Hence rejection sampling returns consistent posterior estimates.

Problem: hopelessly expensive if \( P(e) \) is small.

\( P(e) \) drops off exponentially with number of evidence variables!
Likelihood weighting

Idea: fix evidence variables, sample only nonevidence variables, and weight each sample by the likelihood it accords the evidence

```plaintext
function Likelihood-Weighting(X, e, bn, N) returns an estimate of P(X|e)
    local variables: W, a vector of weighted counts over X; initially zero
    for j = 1 to N do
        x, w ← Weighted-Sample(bn)
        W[x] ← W[x] + w where x is the value of X in x
    return Normalize(W[X])

function Weighted-Sample(bn, e) returns an event and a weight
    x ← an event with n elements; w ← 1
    for i = 1 to n do
        if X_i has a value x_i in e
            then w ← w × P(X_i = x_i | Parents(X_i))
        else x_i ← a random sample from P(X_i | Parents(X_i))
    return x, w
```
Likelihood weighting example

$w = 1.0$
Likelihood weighting example

- **P(C)**: 0.50
- **P(R|C)**:
  - T: 0.80
  - F: 0.20
- **P(S|C)**:
  - T: 0.10
  - F: 0.50
- **P(W|S,R)**:
  - TT: 0.99
  - TF: 0.90
  - FT: 0.90
  - FF: 0.01

Diagram:
- Cloudy
- Rain
- Sprinkler
- Wet Glass
Likelihood weighting example

\[ w = 1.0 \]
Likelihood weighting example

\[
P(C) = 0.50
\]

\[
\begin{array}{|c|c|}
\hline
C & P(S|C) \\
\hline
T & 0.10 \\
F & 0.50 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
C & P(R|C) \\
\hline
T & 0.80 \\
F & 0.20 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
S & R & P(W|S,R) \\
\hline
T & T & 0.99 \\
T & F & 0.90 \\
F & T & 0.90 \\
F & F & 0.01 \\
\hline
\end{array}
\]

\[w = 1.0 \times 0.1\]
Likelihood weighting example

\[ w = 1.0 \times 0.1 \times 0.99 = 0.099 \]
Some more examples

- Farming, Microsoft Trouble Shooters etc.
- Most are diagnosis based.
- Useful when we want to reason in the presence of uncertainty. Next lectures. Chapter 7, reasoning in the presence of certainty using propositional logic.
Knowledge Based Agents

Belief Networks

Prediction/Diagnosis Graph

Knowledge base = set of sentences in a formal language

Same as Before

Declarative approach to building an agent (or other system):

TELL it what it needs to know

Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level
i.e., what they know, regardless of how implemented

Or at the implementation level
i.e., data structures in KB and algorithms that manipulate them

Consider the elevator control problem
Knowledge Base Agent Wrapper

```
function KB-AGENT(percept) returns an action
    static: KB, a knowledge base
            t, a counter, initially 0, indicating time
    TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
    action ← ASK(KB, MAKE-ACTION-QUERY(t))
    TELL(KB, MAKE-ACTION-SENTENCE(action, t))
    t ← t + 1
    return action
```

The agent must be able to:
Represent states, actions, etc.
Incorporate new percepts
Update internal representations of the world
Deduce hidden properties of the world
Deduce appropriate actions

Propositional and first order logic is just one way to represent Knowledge and reason what actions to perform. Typically no optimization
Logics at High Level

Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the "meaning" of sentences; i.e., define truth of a sentence in a world.

E.g., the language of arithmetic

\( x + 2 \geq y \) is a sentence; \( x^2 + y > \) is not a sentence.

\( x + 2 \geq y \) is true iff the number \( x + 2 \) is no less than the number \( y \).

\( x + 2 \geq y \) is true in a world where \( x = 7, \ y = 1 \).

\( x + 2 \geq y \) is false in a world where \( x = 0, \ y = 6 \).
Notion of a Model: Model is an “assignment of literals to propositions”

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \)

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

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

E.g. \( KB = \) Giants won and Reds won
\( \alpha = \) Giants won
\( KB = A \cap B \cap C, M(KB):TTT \)
\( \alpha = A \cap B, M(\alpha): TTF, TTT \)
KB entails \( \alpha \)?
KB \( = A \cup B, M(KB): ??? \)
\( \alpha = A, M(\alpha): ??? KB \) entails \( \alpha ??? \)
Inference Algorithm Properties

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

**Soundness:** $i$ is sound if

whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

**Completeness:** $i$ is complete if

whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Logic Types

Logics are characterized by what they commit to as “primitives”

**Ontological commitment:** what exists—facts? objects? time? beliefs?

**Epistemological commitment:** what states of knowledge?

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0…1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0…1</td>
</tr>
</tbody>
</table>

Compare to belief networks
Propositional Logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols $P_1$, $P_2$ etc are sentences

If $S$ is a sentence, $\neg S$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \land S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \lor S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Rightarrow S_2$ is a sentence

If $S_1$ and $S_2$ is a sentence, $S_1 \Leftrightarrow S_2$ is a sentence

Interesting trade-off between language complexity and computation
Propositional Logic: Semantics

Each model specifies true/false for each proposition symbol

E.g. \( A \quad B \quad C \)

\[
\begin{array}{l}
\text{True} & \text{True} & \text{False}
\end{array}
\]

Rules for evaluating truth with respect to a model \( m \):

\[
\begin{align*}
\neg S & \quad \text{is true iff} \quad S & \quad \text{is false} \\
S_1 \land S_2 & \quad \text{is true iff} \quad S_1 & \quad \text{is true and} \quad S_2 & \quad \text{is true} \\
S_1 \lor S_2 & \quad \text{is true iff} \quad S_1 & \quad \text{is true or} \quad S_2 & \quad \text{is true} \\
S_1 \Rightarrow S_2 & \quad \text{is true iff} \quad S_1 \quad \text{i.e., is false iff} \quad S_1 & \quad \text{is true and} \quad S_2 & \quad \text{is false} \\
S_1 \Leftrightarrow S_2 & \quad \text{is true iff} \quad S_1 \Rightarrow S_2 & \quad \text{is true and} \quad S_2 \Rightarrow S_1 & \quad \text{is true}
\end{align*}
\]
Inference via Enumeration

Let \( \alpha = A \lor B \) and \( KB = (A \lor C) \land (B \lor \neg C) \)

Is it the case that \( KB \models \alpha \)?
Check all possible models—\( \alpha \) must be true wherever \( KB \) is true
Performance measure
  gold +1000, death -1000
  -1 per step, -10 for using the arrow

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 Breeze, Glitter, Smell

Actuators Left turn, Right turn,
  Forward, Grab, Release, Shoot
Situation after detecting nothing in \([1,1]\), moving right, breeze in \([2,1]\).

Consider possible models for \(\square\)s assuming only pits

\(3\) Boolean choices \(\Rightarrow\) \(8\) possible models
\( KB = \text{wumpus-world rules + observations} \)

\( KB = \text{wumpus-world rules + observations} \)

\( \alpha_1 = \text{"[1,2] is safe", } KB \models \alpha_1, \text{ proved by model checking} \)
Mine Sweeper

- Design a logical agent to play minesweeper
Propositional Logic Representation

- Let $X_{i,j}$ be true iff location $[i,j]$ contains a mine.
- “A location adjacent to $[3,4]$ contains a mine”
- “There is exactly one mine adjacent to location $[3,4]$”
Propositional Logic Representation

• Let \( X_{i,j} \) be true iff location \([i,j]\) contains a mine.

• “A location adjacent to \([3,4]\) contains a mine”
  \[ X_{2,3} \lor X_{2,4} \lor X_{2,5} \lor X_{3,3} \lor X_{3,5} \lor X_{4,3} \lor X_{4,4} \lor X_{4,5} \]

• “There is exactly one mine adjacent to location \([3,4]\)”
  \[ (X_{2,3} \land \neg X_{2,4} \land \neg X_{2,5} \land \neg X_{3,3} \land \neg X_{3,5} \land \neg X_{4,3} \land \neg X_{4,4} \land \neg X_{4,5}) \lor \land \]
Inference

• Suppose location [1, 1] has exactly one adjacent mine, and [1, 2] is marked with a mine.
  – KB entails \( X_{2,1} \)?

• KB

\[
R_1 : X_{1,2}
\]
\[
R_2 : (X_{1,2} \land \neg X_{2,1} \land \neg X_{2,2}) \lor (\neg X_{1,2} \land X_{2,1} \land \neg X_{2,2}) \lor (\neg X_{1,2} \land \neg X_{2,1} \land X_{2,2})
\]
Inference

- Enumerate all possible models,
  - KB is true (i.e. every sentence is true) iff sentence is true

<table>
<thead>
<tr>
<th>$X_{1,2}$</th>
<th>$X_{2,1}$</th>
<th>$X_{2,2}$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>KB</th>
<th>$\neg X_{2,1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
Notion of a Model

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$.

$M(\alpha)$ is the set of all models of $\alpha$.

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

E.g. $KB = \text{Giants won and Reds won}$

$\alpha = \text{Giants won}$
Okay. We can perform inference using model checking. But what’s the computational problem?

Is there a better way
Okay. We can perform inference using model checking. But what’s the computational problem? Is there a better way? Yes. 1) Limit formal language. 2) Uninformed search technique and rules of inference. 3) Use universal inference technique.
Logical equivalence

Two sentences are logically equivalent iff true in same models:
\[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

\[ (\alpha \land \beta) \equiv (\beta \land \alpha) \text{ commutativity of } \land \]
\[ (\alpha \lor \beta) \equiv (\beta \lor \alpha) \text{ commutativity of } \lor \]
\[ ((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \text{ associativity of } \land \]
\[ ((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \text{ associativity of } \lor \]
\[ \neg(\neg \alpha) \equiv \alpha \text{ double-negation elimination} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \text{ contraposition} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \text{ implication elimination} \]
\[ (\alpha \leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \text{ biconditional elimination} \]
\[ \neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \text{ de Morgan} \]
\[ \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \text{ de Morgan} \]
\[ (\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \text{ distributivity of } \land \text{ over } \lor \]
\[ (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \text{ distributivity of } \lor \text{ over } \land \]
Horn Clauses

• Disjunctions with at most one positive literal. I.e. 「Thunder ∪ 「Rain ∪ Lightening

• Rewrite as
Horn Clauses

- Disjunctions with at most one positive literal. I.e. Thunder ∪ Rain ∪ Lightening
- Rewrite as:
  - Rain ∩ Thunder => Lightening
  - Knowledge Base
    - Rain ∩ Thunder => Lightening
    - Cloudy ∩ PressureChange => Rain
    - Cloudy
    - PressureChange
Modus Ponens

• eliminates =>

\[(X \Rightarrow Y), \quad X\]

\[\underline{__________}\]

\[Y\]

– If it rains, then the streets will be wet.
– It is raining.
– Infer the conclusion: The streets will be wet.
  (affirms the antecedent)
Inferencing

- **Forward Chaining**
  - Start with the knowledge base and through repeated applications of Modus Ponens, derive all atomic sentences.
  - Data driven

- **Backward Chaining**
  - Start with the query $q$
  - Find implications that conclude (i.e. have head) $q$
  - If all premises are true, then $q$ is true, otherwise use backward chaining on premises with unknown values.
  - Goal-directed
But of course this is very limiting. PL is very simple to begin with and we are using a restriction.
Exercise

• “If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.”

• Can you prove it’s mythical? Magical? Horned?
Solution

1. \((\text{Mythical} \Rightarrow \text{Immortal}) \land (\neg \text{Mythical} \Rightarrow \text{Mortal} \land \text{Mammal})\)
2. \((\text{Immortal} \lor \text{Mammal} \Rightarrow \text{Horned})\)
3. \((\text{Horned} \Rightarrow \text{Magical})\)
4. \(\text{Mythical} \Rightarrow \text{Immortal} \quad \text{AE}(1)\)
5. \(\neg \text{Mythical} \Rightarrow \text{Mortal} \land \text{Mammal} \quad \text{AE}(1)\)
6. \(\neg \text{Mythical} \lor \text{Immortal} \quad \text{IE}(4)\)
7. \(\neg \neg \text{Mythical} \lor (\text{Mortal} \land \text{Mammal}) \quad \text{IE}(5)\)
8. \(\text{Immortal} \lor (\text{Mortal} \land \text{Mammal}) \quad \text{Resolution}(6,7)\)
9. \((\text{Immortal} \lor \text{Mortal}) \land (\text{Immortal} \lor \text{Mammal}) \quad \text{DL}(8)\)
10. \((\text{Immortal} \lor \text{Mammal}) \quad \text{AE}(9)\)
11. \(\text{Horned} \quad \text{MP}(2,10)\)
12. \(\text{Magical} \quad \text{MP}(3,11)\)
Logical equivalence

Two sentences are logically equivalent iff true in same models:
\[ \alpha \equiv \beta \quad \text{if and only if} \quad \alpha \models \beta \text{ and } \beta \models \alpha \]

\[ (\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \]
\[ (\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \]
\[ (((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \]
\[ (((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \]
\[ \neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \]
\[ (\alpha \iff \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \]
\[ \neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \]
\[ \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \]
\[ (\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \]
\[ (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land \]
2nd Lecture on P.L.
Notion of a Model:
Model is an “assignment of literals to propositions”

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$.

$M(\alpha)$ is the set of all models of $\alpha$.

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

E.g. $KB = \text{Giants won and Reds won}$
$\alpha = \text{Giants won}$
$KB = A \cap B \cap C$, $M(KB):\text{TTT}$
$\alpha = A \cap B$, $M(\alpha): \text{TTF, TTT}$
$KB$ entails $\alpha$?

$KB = A \cup B$, $M(KB):??$
$\alpha = A$, $M(\alpha): ??$ $KB$ entails $\alpha$??
Inference via Enumeration

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?
Check all possible models—$\alpha$ must be true wherever $KB$ is true

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$A \lor C$</th>
<th>$B \lor \neg C$</th>
<th>$KB$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td></td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>
Inference

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

Soundness: $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Upto Here
The Big Picture - 1

- We can create intelligent behavior by say search as in A* but that does have limitations.
- We can create more general intelligent behavior by reasoning from a set of knowledge. But to reason you need to represent the knowledge.
- Belief networks were one such approach allowing us to state the **probabilistic** cause-effect relationship between events.
- Logics are another way …
- Begin with propositional logic then onto first order.
Reasoning: Propositional Logic

• Set up a knowledge base containing sentences. These sentences state facts that we know with certainty.

• We can then reason/infer what other facts will occur (with certainty).

• We reason that sentence \( x \) must occur given the knowledge base is true if KB entails \( x \).
Inference

• Suppose location [1,1] has exactly one adjacent mine, and [1,2] is marked with a mine.
  – KB entails $X_{2,1}$?

• KB

\[ R_1 : X_{1,2} \]
\[ R_2 : (X_{1,2} \land \neg X_{2,1} \land \neg X_{2,2}) \lor (\neg X_{1,2} \land X_{2,1} \land \neg X_{2,2}) \lor (\neg X_{1,2} \land \neg X_{2,1} \land X_{2,2}) \]

• But we want an algorithm to automatically reason for us …
Inference Algorithm Properties

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

**Soundness:** $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

**Completeness:** $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Validity and Satisfiability

A sentence is valid if it is true in all models,
   e.g.,

Validity is connected to inference via the Deduction Theorem:
   \[ KB \models \alpha \text{ if and only if } (KB \Rightarrow \alpha) \]

A sentence is satisfiable if it is true in some model
   e.g.,

A sentence is unsatisfiable if it is true in no models
   e.g.,

Satisfiability is connected to entailment via the following:
   how ???

What proof technique is this using
Validity and satisfiability

A sentence is valid if it is true in all models,
  e.g., True, A \lor \neg A, A \Rightarrow A, (A \land (A \Rightarrow B)) \Rightarrow B

Validity is connected to inference via the Deduction Theorem:
  KB \models \alpha if and only if (KB \Rightarrow \alpha) is valid

A sentence is satisfiable if it is true in some model
  e.g., A \lor B, C

A sentence is unsatisfiable if it is true in no models
  e.g., A \land \neg A

Satisfiability is connected to inference via the following:
  KB \models \alpha if and only if (KB \land \neg \alpha) is unsatisfiable

What proof technique is this using
Let's Examine Proof by Contradiction

- **Theorem.** There are infinitely many prime numbers.
- **Proof.** Assume to the contrary that there are only finitely many prime numbers, and all of them are listed as follows: $p_1, p_2, ..., p_n$. Consider the number $q = p_1 p_2 ... p_n + 1$. This number is not divisible by any of the listed primes since if we divided $p_i$ into $q$, there would result a remainder of 1 for each $i = 1, 2, ..., n$. Well then, we must conclude that $q$ is a prime number, not among the primes listed above, contradicting our assumption that all primes are in the list $p_1, p_2, ..., p_n$.
- We’ll use proof by contradiction in more advance inference techniques.
Validity, satisfiability, and unsatisfiability are properties of individual sentences.

In logical reasoning, we are not so much concerned with individual sentences as we are with the relationships between sentences. In particular, we would like to know, given some sentences, whether other sentences are or are not logical conclusions. This relative property is known as *logical entailment*. When we are speaking about Propositional Logic, we use the phrase *propositional entailment*.

A set of sentences $\Delta$ *logically entails* a sentence $\varphi$ (written $\Delta |\!|= \varphi$) if and only if every interpretation/model/world that satisfies $\Delta$ also satisfies $\varphi$.
Notion of a Model

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \).

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

Then \( KB \models \alpha \) if and only if \( M(KB) \subseteq M(\alpha) \).

E.g. \( KB = \) Giants won and Reds won

\( \alpha = \) Giants won
Reasoning: Belief Networks vs Propositional Logic

• What can be represented?
• Then how do they differ
• Other more subtle differences
  – No notion of reasoning away with PL.
Inference via Enumeration (Model Checking)

Let $\alpha = A \lor B$ and $KB = (A \lor C) \land (B \lor \neg C)$

Is it the case that $KB \models \alpha$?
Check all possible models—$\alpha$ must be true wherever $KB$ is true

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
<td>$C$</td>
<td>$A \lor C$</td>
<td>$B \lor \neg C$</td>
<td>$KB$</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Logical equivalence

Two sentences are logically equivalent iff true in same models:
\[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

\[(\alpha \land \beta) \equiv (\beta \land \alpha) \text{ commutativity of } \land\]
\[(\alpha \lor \beta) \equiv (\beta \lor \alpha) \text{ commutativity of } \lor\]
\[((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \text{ associativity of } \land\]
\[((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \text{ associativity of } \lor\]
\[\neg(\neg \alpha) \equiv \alpha \text{ double-negation elimination}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \text{ contraposition}\]
\[(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \text{ implication elimination}\]
\[(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \text{ biconditional elimination}\]
\[\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \text{ de Morgan}\]
\[\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \text{ de Morgan}\]
\[(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \text{ distributivity of } \land \text{ over } \lor\]
\[(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \text{ distributivity of } \lor \text{ over } \land\]
Exercise not Morbid Concepts …

• “If the unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical if it is horned.”

• Can you prove it’s mythical? Magical? Horned?
Solution

1. \((\text{Mythical} \Rightarrow \text{Immortal}) \land (\neg \text{Mythical} \Rightarrow \text{Mortal} \land \text{Mammal})\)
2. \((\text{Immortal} \lor \text{Mammal} \Rightarrow \text{Horned})\)
3. \((\text{Horned} \Rightarrow \text{Magical})\)
4. \(\text{Mythical} \Rightarrow \text{Immortal} \quad \text{AE(1)}\)
5. \((\neg \text{Mythical} \Rightarrow \text{Mortal} \land \text{Mammal}) \quad \text{AE(1)}\)
6. \((\neg \text{Mythical} \lor \text{Immortal}) \quad \text{IE(4)}\)
7. \((\neg \neg \text{Mythical} \lor (\text{Mortal} \land \text{Mammal})) \quad \text{IE(5)}\)
8. \((\text{Immortal} \lor (\text{Mortal} \land \text{Mammal})) \quad \text{Resolution(6,7)}\)
9. \((\text{Immortal} \lor \text{Mortal}) \land (\text{Immortal} \lor \text{Mammal}) \quad \text{DL(8)}\)
10. \((\text{Immortal} \lor \text{Mammal}) \quad \text{AE(9)}\)
11. \(\text{Horned} \quad \text{MP(2,10)}\)
12. \(\text{Magical} \quad \text{MP(3,11)}\)
Uninformed Search

• Could use BFS or DFS to solve this problem. How?
Horn Clauses

• Model checking takes exponential time as can uninformed search.
• If we limit the sentences we can construct, then we can get polynomial time algorithms.
• Disjunctions with at most one positive literal. I.e. 「Thunder ∪ 「Rain ∪ Lightening
• Rewrite as
Horn Clauses

• Disjunctions with at most one positive literal. I.e. 
  ¬Thunder ∪ ¬Rain ∪ Lightening
• Rewrite as:
• Rain ∩ Thunder => Lightening
• Knowledge Base
  – Rain ∩ Thunder => Lightening
  – Cloudy ∩ PressureChange => Rain
  – Cloudy
  – PressureChange
Modus Ponens

• eliminates =>
  \[(X \implies Y), \quad X\]

\[
\underline{Y}
\]

– If it rains, then the streets will be wet.
– It is raining.
– Infer the conclusion: The streets will be wet.
  (affirms the antecedent)
Inferencing

- **Forward Chaining**
  - Start with the knowledge base and through repeated applications of Modus Ponens, derive all atomic sentences.
  - Data driven

- **Backward Chaining**
  - Start with the query $q$
  - Find implications that conclude (i.e. have head) $q$
  - If all premises are true, then $q$ is true, otherwise use backward chaining on premises with unknown values.
  - Goal-directed
Forward chaining

- Idea: fire any rule whose premises are satisfied in the KB,
  - add its conclusion to the KB, until query is found

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A \\
B
\end{align*}
\]
Forward chaining algorithm

function PL-FC-ENTAILS?(KB, q) returns true or false

local variables: count, a table, indexed by clause, initially the number of premises
inferred, a table, indexed by symbol, each entry initially false
agenda, a list of symbols, initially the symbols known to be true

while agenda is not empty do
    p ← Pop(agenda)
    unless inferred[p] do
        inferred[p] ← true
        for each Horn clause c in whose premise p appears do
            decrement count[c]
            if count[c] = 0 then do
                if HEAD[c] = q then return true
                PUSH(HEAD[c], agenda)
        end for
    end unless
end while

return false

• Forward chaining is sound and complete for Horn KB
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Backward chaining

Idea: work backwards from the query $q$: 

to prove $q$ by BC,
  check if $q$ is known already, or
  prove by BC all premises of some rule concluding $q$

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
  1. has already been proved true, or
  2. has already failed
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Backward chaining example
Forward vs. backward chaining

• FC is data-driven, automatic, unconscious processing,
  – e.g., object recognition, routine decisions

• May do lots of work that is irrelevant to the goal

• BC is goal-driven, appropriate for problem-solving,
  – e.g., Where are my keys? How do I get into a PhD program?

• Complexity of BC can be much less than linear in size of KB
Logical equivalence

Two sentences are logically equivalent iff true in same models:

\[ \alpha \equiv \beta \quad \text{if and only if} \quad \alpha \models \beta \text{ and } \beta \models \alpha \]

\[
\begin{align*}
(\alpha \land \beta) &\equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) &\equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) &\equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) &\equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
(\neg \neg \alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
(\neg (\alpha \land \beta)) &\equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
(\neg (\alpha \lor \beta)) &\equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) &\equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) &\equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land
\end{align*}
\]
Up to Here

Today: Wrap up PL
Revise forward-chaining and resolution
Summarize all four techniques limits
Compare reasoning with PL and BN when to use each
Notion of a Model

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in $m$.

$M(\alpha)$ is the set of all models of $\alpha$.

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

E.g. $KB = \text{Giants won and Reds won}$

$\alpha = \text{Giants won}$
Inference Algorithm Properties

$KB \vdash_i \alpha$ = sentence $\alpha$ can be derived from $KB$ by procedure $i$

Soundness: $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
Validity and satisfiability

A sentence is valid if it is true in all models,
e.g., *True*,  \( A \lor \neg A \),  \( A \Rightarrow A \),  \( (A \land (A \Rightarrow B)) \Rightarrow B \)

Validity is connected to inference via the Deduction Theorem:
\[ KB \models \alpha \text{ if and only if } (KB \Rightarrow \alpha) \text{ is valid} \]

A sentence is satisfiable if it is true in some model
e.g., \( A \lor B \),  \( C \)

A sentence is unsatisfiable if it is true in no models
e.g., \( A \land \neg A \)

Satisfiability is connected to inference via the following:
\[ KB \models \alpha \text{ if and only if } (KB \land \neg \alpha) \text{ is unsatisfiable} \]

What proof technique is this using
Horn Clauses

• Model checking takes exponential time as can uninformed search.
• If we limit the sentences we can construct, then we can get polynomial time algorithms.
• Disjunctions with at most one positive literal. I.e. 「Thunder ∪ 「Rain ∪ Lightening
• Rewrite as
Modus Ponens

- eliminates =>
  \[(X \implies Y),\ X\]
  \[\implies Y\]
  - If it rains, then the streets will be wet.
  - It is raining.
  - Infer the conclusion: The streets will be wet. (affirms the antecedent)
  - Let’s examine this rule in more detail
Inferencing

- **Forward Chaining**
  - Start with the knowledge base and through repeated applications of Modus Ponens, derive all atomic sentences.
  - Data driven

- **Backward Chaining**
  - Start with the query $q$
  - Find implications that conclude (i.e. have head) $q$
  - If all premises are true, then $q$ is true, otherwise use backward chaining on premises with unknown values.
  - Goal-directed
Forward chaining

- Idea: fire any rule whose premises are satisfied in the $KB$, add its conclusion to the $KB$, until query is found

\[
\begin{align*}
P & \Rightarrow Q \\
L \land M & \Rightarrow P \\
B \land L & \Rightarrow M \\
A \land P & \Rightarrow L \\
A \land B & \Rightarrow L \\
A \\
B
\end{align*}
\]
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
Forward chaining example
We don’t need all these rules, we only need one inference rule!!!!
Conjunctive Normal Form (CNF—universal)

conjunction of disjunctions of literals

clauses

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Resolution inference rule (for CNF): complete for propositional logic

\[
\begin{array}{c}
\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n \\
\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n \\
\end{array}
\]

where \(\ell_i\) and \(m_j\) are complementary literals. E.g.,

\[
\frac{P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}
\]

Resolution is sound and complete for propositional logic
### Propositional Resolution Example

**Prove R**

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$P \lor Q$</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>$\neg P \lor R$</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>$\neg Q \lor R$</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>$\neg R$</td>
<td>Negated conclusion</td>
</tr>
</tbody>
</table>

$\neg R$ \\
$\rightarrow R \lor \text{false}$ \\
$\text{false} \lor \text{false}$
### Propositional Resolution Example

**Prove R**

<table>
<thead>
<tr>
<th>Step</th>
<th>Formula</th>
<th>Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( P \lor Q )</td>
<td>Given</td>
</tr>
<tr>
<td>2</td>
<td>( \neg P \lor R )</td>
<td>Given</td>
</tr>
<tr>
<td>3</td>
<td>( \neg Q \lor R )</td>
<td>Given</td>
</tr>
<tr>
<td>4</td>
<td>( \neg R )</td>
<td>Negated conclusion</td>
</tr>
<tr>
<td>5</td>
<td>( Q \lor R )</td>
<td>1, 2</td>
</tr>
<tr>
<td>6</td>
<td>( \neg P )</td>
<td>2, 4</td>
</tr>
<tr>
<td>7</td>
<td>( \neg Q )</td>
<td>3, 4</td>
</tr>
<tr>
<td>8</td>
<td>( R )</td>
<td>5, 7</td>
</tr>
<tr>
<td>9</td>
<td>( \cdot )</td>
<td>4, 8</td>
</tr>
</tbody>
</table>

\( \text{false} \lor R \)

\( \neg R \lor \text{false} \)

\( \text{false} \lor \text{false} \)
Resolution example

\[ KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} \alpha = \neg P_{1,2} \]
Conversion to CNF

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)\).

\[ (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]

3. Move \( \neg \) inwards using de Morgan's rules and double-negation:

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1}) \]

4. Apply distributivity law \( (\lor \text{ over } \land) \) and flatten:

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1}) \]
Logical equivalence

Two sentences are logically equivalent iff true in same models:
\[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

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

\[ \neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \]
\[ (\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \]
\[ (\alpha \iff \beta) \equiv (((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \]
\[ \neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \]
\[ \neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \]
\[ (\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \]
\[ (\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land \]
Resolution algorithm

Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

function PL-RESOLUTION($KB, \alpha$) returns true or false

  clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \land \neg \alpha$
  new $\leftarrow \{\}$
  loop do
    for each $C_i, C_j$ in clauses do
      resolvents $\leftarrow$ PL-RESOLVE($C_i, C_j$)
      if resolvents contains the empty clause then return true
      new $\leftarrow$ new $\cup$ resolvents
      if new $\subseteq$ clauses then return false
      clauses $\leftarrow$ clauses $\cup$ new
Inference Algorithm Properties

$KB \vdash_i \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$

**Soundness:** $i$ is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

**Completeness:** $i$ is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
So four approaches to perform inference:

1) Model checking
   Sound, complete? Limitation

2) FW/BW chaining
   Sound, complete? Limitation

3) Uninformed search and all previous rules
   Sound, complete? Limitation

4) Resolution
   Sound, complete? Limitation
Comparison Between PL and Belief Networks

Let’s state the sentences equivalent to each graph

(a)  
(b)  
(c)
Comparison Between PL and Belief Networks

(a) Y→X, Z→X
(b) X→Y∧Z
(c) Y→X, X→Z

Now let's recall how Y effects Z when X is observed or not and see whether similar behavior occurs in PL
Comparison Between PL and Belief Networks

Now let's recall how Y affects Z when X is observed or not and see whether similar behavior occurs in PL.

(a) Observed: Z and Y dependent
    Unobserved: no uncond. Indep.
(b) Observed: cond. independent,
    Unobserved: independent
(c) Observed: cond. Independent
    Unobserved: dependent
Comparison Between PL and Belief Networks

(a) Observed: Z and Y dependent
Unobserved: uncond. Indep.

(b) Observed: cond. independent,
Unobserved: independent

(c) Observed: cond. Independent
Unobserved: dependent

(a) Y → X, Z → X
Obs: X, Y, Z occur?
Unobs: Y occurs?

(b) X → Y, X → Z
Obs: Y and X occur Z occurs regardless of Y
Unobs: Knowledge of Y tells me nothing of Z

(c) Y → X, X → Z
Obs: X and Y I will know the value of Z
Unobs: Y occurs, then X occurs and then Z occurs
Complete equivalent in PL, Ignoring probabilities
Example

\[ C \Rightarrow S, \quad C \Rightarrow R, \quad S \land R \Rightarrow WG \]
\[ WG \land \neg S \Rightarrow R \]
\[ WG \land \neg R \Rightarrow S \]
C⇒S, C⇒R, S∧R⇒WG

WG ∧¬S ⇒R
WG ∧¬R ⇒S
WG
S
KB entail R?
Entailment

1: C=>S, 2: C=>R, 3: S∧R=>WG
4: WG ∧¬S =>R, 5: WG ∧¬R =>S
6: WG, 7: S
KB entail R?

What inference technique can we use?
Entailment

1: C=>S, 2: C=>R, 3: S \land R=>WG
4: WG \land \neg S => R, 5: WG \land \neg R => S
6: WG, 7: S
KB entail R?

Lets use resolution and proof by negation
By adding in 8: \neg R and trying to find a contradiction
First we must convert the KB into conjunctive normal form.

1: \neg C \lor S, 2: \neg C \lor R, 3: \neg (S \land R) \lor WG \equiv \neg S \lor \neg R \lor WG
4: WG \land \neg S => R \equiv \neg (WG \land \neg S) \lor R \equiv \neg WG \lor S \lor R,
5: \neg WG \lor S \lor R, 6: WG, 7: S, 8: \neg R
Entailment

1: \(\neg C \lor S\), 2: \(\neg C \lor R\), 3: \(\neg S \lor \neg R \lor WG\)
4: \(\neg WG \lor S \lor R\),
5: \(\neg WG \lor S \lor R\), 6: WG, 7: S, 8: \(\neg R\)

We should try to show a contradiction with S or WG
Resolve 8 with 2 to get 9: \(\neg C\)
Resolve 7 with 3 to get 10: \(\neg R \lor WG\)
Resolve 4 with 10: to get S no contradiction!!!
**Logical equivalence**

Two sentences are logically equivalent iff true in same models:

\[ \alpha \equiv \beta \text{ if and only if } \alpha \models \beta \text{ and } \beta \models \alpha \]

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \text{ commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \text{ commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \text{ associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \text{ associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \text{ double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \text{ contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \text{ implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \text{ biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \text{ de Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \text{ de Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \text{ distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \text{ distributivity of } \lor \text{ over } \land
\end{align*}
\]
Problems with PL Agents

• As board size increases, the number of propositions increase dramatically
• Building the sentences for bombs in surrounding squares is tedious
• Consider situations with agents in time:

\[ L_{1,1} \land \text{FacingSout} h \land \text{Forward} \Rightarrow L_{1,2} \]

\[ L_{1,1}^1 \land \text{FacingSout} h^1 \land \text{Forward}^1 \Rightarrow L_{1,2}^2 \]
Pros and cons of propositional logic

Propositional logic is declarative: pieces of syntax correspond to facts.

Propositional logic allows partial/disjunctive/negated information
(Unlike most data structures and databases)

Propositional logic is compositional:
meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$

Meaning in propositional logic is context-independent
(Unlike natural language, where meaning depends on context)

Propositional logic has very limited expressive power
(Unlike natural language)
E.g., cannot say “pits cause breezes in adjacent squares” except by writing one sentence for each square

MineSweeper situation: Single Mine Around 1,1
First Order Logic overcomes this problem, but at several costs
Inference

\( KB \vdash_i \alpha = \) sentence \( \alpha \) can be derived from \( KB \) by procedure \( i \)

**Soundness:** \( i \) is sound if

whenever \( KB \vdash_i \alpha \), it is also true that \( KB \models \alpha \)

**Completeness:** \( i \) is complete if

whenever \( KB \models \alpha \), it is also true that \( KB \vdash_i \alpha \)
Validity and satisfiability

A sentence is valid if it is true in all models,
e.g., True, A ∨ ¬A, A ⇒ A, (A ∧ (A ⇒ B)) ⇒ B

Validity is connected to inference via the Deduction Theorem:
KB ⊨ α if and only if (KB ⇒ α) is valid

A sentence is satisfiable if it is true in some model
e.g., A ∨ B, C

A sentence is unsatisfiable if it is true in no models
e.g., A ∧ ¬A

Satisfiability is connected to inference via the following:
KB ⊨ α if and only if (KB ∧ ¬α) is unsatisfiable

What proof technique is this using
Okay. We can perform inference using model checking. But what’s the computational problem?

Is there a better way

Yes. 1) Limit formal language
2) Uninformed search technique
and rules of inference
3) Use universal inference technique
Notion of a Model: Model is an “assignment of literals to propositions”

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

We say \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \).

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

Then \( KB \models \alpha \) if and only if \( M(KB) \subseteq M(\alpha) \).

E.g. \( KB = \) Giants won and Reds won

\( \alpha = \) Giants won

\( KB = A \cap B \cap C, \ M(KB):TTT \)

\( \alpha = A \cap B, \ M(\alpha): TTF, TTT \)

\( KB \) entails \( \alpha \)??

\( KB = A \cup B, \ M(KB):?? \)

\( \alpha = A, \ M(\alpha): ??? \) KB entails \( \alpha \)??
Is entailment:
1) Implication
2) Equivalence
Propositional Entailment

Validity, satisfiability, and unsatisfiability are properties of individual sentences. In logical reasoning, we are not so much concerned with individual sentences as we are with the relationships between sentences. In particular, we would like to know, given some sentences, whether other sentences are or are not logical conclusions. This relative property is known as \textit{logical entailment}. When we are speaking about Propositional Logic, we use the phrase \textit{propositional entailment}.

A set of sentences $\Delta$ \textit{logically entails} a sentence $\varphi$ (written $\Delta \models \varphi$) if and only if every interpretation/model/world that satisfies $\Delta$ also satisfies $\varphi$. 
Stating Facts about the World is Cumbersome in PL

i.e. MineSweeper
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, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
- Relations: red, round, bogus, prime, multistoried ..., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
- Functions: father of, best friend, third inning of, one more than, beginning of ...

Note functions are effectively one-one mappings so return a unique object. Relate a given object to precisely one object. F.O.L. powerful representation language, but entailment becomes undecidable or does it …
Syntax of FOL: Basic elements

Constants: KingJohn, 2, UCB, ...
Predicates: Brother, >, ...
Functions: sqrt, LeftLegOf, ...
Variables: x, y, a, b, ...
Connectives: ∧, ∨, →, ⇒, ↔
Equality: =
Quantifiers: ∀, ∃
Atomic sentences

Atomic sentence = predicate(term₁, ..., termₙ)
    or term₁ = term₂

    Term = function(term₁, ..., termₙ)
    or constant or variable

E.g., Brother(KingJohn, RichardTheLionheart)
    > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))
**Atomic sentences**

Atomic sentence = \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \)

or \( \text{term}_1 = \text{term}_2 \)

Term = \( \text{function}(\text{term}_1, \ldots, \text{term}_n) \)

or constant or variable

E.g., \( \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) \)

\( > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \)

**Diagram:**

- **Brother (KingJohn, RichardTheLionheart)**
  - predicate
  - constant
  - term
  - constant
  - term
- **atomic sentence**

- **> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))**
  - predicate
  - function
  - function
  - constant
  - function
  - function
  - constant
  - term
  - term
- **atomic sentence**
Complex sentences

Complex sentences are made from atomic sentences using connectives

\[ \neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2 \]

E.g. \[ \text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn}) \]
Complex sentences

Complex sentences are made from atomic sentences using connectives:

$$\neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \implies S_2, \quad S_1 \iff S_2$$

E.g. $\text{Sibling}(\text{KingJohn}, \text{Richard}) \implies \text{Sibling}(\text{Richard}, \text{KingJohn})$
Semantics in First-order Logic

Models of first-order logic

Sentences are true or false with respect to models, which consist of

- a domain (also called universe)
- an interpretation

Domain

A non-empty (finite or infinite) set of arbitrary elements

Interpretation

Assigns to each

- constant symbol: a domain element
- predicate symbol: a relation on the domain (of appropriate arity)
- function symbol: a function on the domain (of appropriate arity)
Models for First-order Logic: Example

Objects, Binary Rels, Unary Rels, Unary functions?
Models for First-order Logic: Example

5 Objects, 2 Binary Rels, 3 Unary Rels, 1 Unary functions?

Colloquially:
Difference between a Function and unary relation
Models for FOL: Lots!

We can enumerate the models for a given KB vocabulary:

For each number of domain elements $n$ from 1 to $\infty$
  For each $k$-ary predicate $P_k$ in the vocabulary
    For each possible $k$-ary relation on $n$ objects
      For each constant symbol $C^i$ in the vocabulary
        For each choice of referent for $C^i$ from $n$ objects ...

Computing entailment by enumerating models is not going to be easy!
Universal Quantification: Syntax

Syntax

\( \forall \text{ variables} \  \text{sentence} \)

Example

\( \forall x \ \text{Studies}(x, \text{SUNY}) \Rightarrow \text{Busy}(x) \)

A common mistake to avoid

Typically, \( \Rightarrow \) is the main connective with \( \forall \)

Common mistake: using \( \land \) as the main connective with \( \forall \):

\( \forall x \ \text{Studies}(x, \text{SUNY}) \land \text{Busy}(x) \)
Existential quantification

\[ \exists \text{(variables)} \ (\text{sentence}) \]

\[ \exists x \ \text{Studies}(x, \text{CSI??}) \land \text{Informed}(x) \]

\[ \exists x \ P \ \text{is true in a model } m \text{ iff } P \text{ with } x \text{ being each possible object in the model} \]

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

Another common mistake to avoid

Typically, \( \land \) is the main connective with \( \exists \)

Common mistake: using \( \Rightarrow \) as the main connective with \( \exists \):

\[ \exists x \ At(x, \text{Stanford}) \Rightarrow \text{Smart}(x) \]
\[ \forall x \ \forall y \ \text{is the same as} \ \forall y \ \forall x \]
\[ \exists x \ \exists y \ \text{is the same as} \ \exists y \ \exists x \]
\[ \exists x \ \forall y \ \text{is not the same as} \ \forall y \ \exists x \]
\[ \exists x \ \forall y \ \text{Loves}(x, y) \]
\[ \forall y \ \exists x \ \text{Loves}(x, y) \]

Quantifier duality: each can be expressed using the other
\[ \forall x \ \text{Likes}(x, \text{IceCream}) \]
\[ \exists x \ \text{Likes}(x, \text{Broccoli}) \]

Translating environmental information into a FOL
\[ \forall x \; \forall y \; \text{is the same as} \; \forall y \; \forall x \]
\[ \exists x \; \exists y \; \text{is the same as} \; \exists y \; \exists x \]
\[ \exists x \; \forall y \; \text{is not the same as} \; \forall y \; \exists x \]
\[ \exists x \; \forall y \; \text{Loves}(x,y) \]
   "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 

\[ \forall x \; \text{Likes}(x, \text{IceCream}) \quad \neg \exists x \; \neg \text{Likes}(x, \text{IceCream}) \]
\[ \exists x \; \text{Likes}(x, \text{Broccoli}) \quad \neg \forall x \; \neg \text{Likes}(x, \text{Broccoli}) \]

Translating environmental information into a FOL
Properties of First-order Logic

Important notions

- validity
- satisfiability
- unsatisfiability
- entailment

are defined for first-order logic in the same way as for propositional logic

Calculi

There are sound and complete calculi for first-order logic (e.g. resolution)

Whenever \( KB \vdash \alpha \), it is also true that \( KB \models \alpha \)

Whenever \( KB \models \alpha \), it is also true that \( KB \vdash \alpha \)

But these calculi CANNOT decide validity, entailment, etc.
Fourth Lecture

Outline
Simple real world problems
Review basics of FOL
Formal languages
Basic inference in FOL
How Can We Represent Situations of This Type in FOL?

A Blocks-World Example

- Two cubes and two pyramids sitting on a table.

Assumptions:

- Cubes may be stacked, and pyramids may be placed atop cubes.
- Only one-on-one stacking is allowed.
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, theories, Ronald McDonald, colors, baseball games, wars, centuries ...
- Relations: red, round, bogus, prime, multistoried ..., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
- Functions: father of, best friend, third inning of, one more than, beginning of ...

Note functions are effectively one-one mappings so return a unique object. Relate a given object to precisely one object. F.O.L. powerful representation language, but entailment becomes undecidable or does it ...
Syntax of FOL: Basic elements

Constants  \textit{KingJohn}, 2, \textit{UCB}, ...  
Predicates  \textit{Brother}, >, ...  
Functions  \textit{Sqrt}, \textit{LeftLegOf}, ...  
Variables  \( x, y, a, b, ... \)  
Connectives  \( \land, \lor, \rightarrow, \Rightarrow, \leftrightarrow \)  
Equality  \( = \)  
Quantifiers  \( \forall, \exists \)
**Atomic sentences**

Atomic sentence = \( predicate(term_1, \ldots, term_n) \)

or \( term_1 = term_2 \)

Term = \( function(term_1, \ldots, term_n) \)

or constant or variable

E.g., \( \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) \)

\( > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \)
Atomic sentences

Atomic sentence = \text{predicate}(term_1, \ldots, term_n)

or \( term_1 = term_2 \)

\[ \text{Term} = \text{function}(term_1, \ldots, term_n) \]

or constant or variable

E.g., \( \text{Brother}(\text{KingJohn}, \text{RichardTheLionheart}) \)

\[ > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \]

\( \text{Brother} ( \text{KingJohn}, \text{RichardTheLionheart} ) \)

\( \text{ predicat} \text{e} \quad \text{constant} \quad \text{constant} \quad \text{term} \quad \text{term} \quad \text{atomic sentence} \)

\[ > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \]

\( \text{predicat} \text{e} \quad \text{function} \quad \text{function} \quad \text{constant} \quad \text{function} \quad \text{function} \quad \text{constant} \quad \text{term} \quad \text{term} \quad \text{atomic sentence} \)
Complex sentences

Complex sentences are made from atomic sentences using connectives

\[ \neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2 \]

E.g. \( \text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn}) \)
Complex sentences are made from atomic sentences using connectives

\[ \neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2 \]

E.g. \( \text{Sibling}(\text{King John}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{King John}) \)
Semantics in First-order Logic

Models of first-order logic

Sentences are true or false with respect to models, which consist of

- a domain (also called universe)
- an interpretation

Domain

A non-empty (finite or infinite) set of arbitrary elements

Interpretation

Assigns to each

- constant symbol: a domain element
- predicate symbol: a relation on the domain (of appropriate arity)
- function symbol: a function on the domain (of appropriate arity)
Objects, Binary Rels, Unary Rels, Unary functions?
Models for First-order Logic: Example

5 Objects, 2 Binary Rels, 3 Unary Rels, 1 Unary function?

Colloquially:
Difference between a Function and unary relation
Models for FOL: Lots!

We can enumerate the models for a given KB vocabulary:

For each number of domain elements $n$ from 1 to $\infty$
  For each $k$-ary predicate $P_k$ in the vocabulary
    For each possible $k$-ary relation on $n$ objects
      For each constant symbol $C$ in the vocabulary
        For each choice of referent for $C$ from $n$ objects . . .

Computing entailment by enumerating models is not going to be easy!
Universal Quantification: Syntax

Syntax

\[ \forall \text{variables} \text{sentence} \]

Example

\[ \forall \ x \ \text{Studies}(x,\text{SUNY}) \Rightarrow \text{Busy}(x) \]

A common mistake to avoid

Typically, \( \Rightarrow \) is the main connective with \( \forall \)

Common mistake: using \( \land \) as the main connective with \( \forall \):

\[ \forall \ x \ \text{Studies}(x,\text{SUNY}) \land \text{Busy}(x) \]
Existential quantification

\( \exists \langle \text{variables} \rangle \langle \text{sentence} \rangle \)

\( \exists x \; \text{Studies}(x, \text{CSI???}) \land \text{Informed}(x) – ??? \)

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

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

Another common mistake to avoid

Typically, \( \land \) is the main connective with \( \exists \)

Common mistake: using \( \Rightarrow \) as the main connective with \( \exists \):

\( \exists x \; \text{At}(x, \text{Stanford}) \Rightarrow \text{Smart}(x) \)
\( \forall x \ \forall y \) 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 \( \forall y \ \exists x \)

\( \exists x \ \forall y \ \text{Loves}(x,y) \)

\( \forall y \ \exists x \ \text{Loves}(x,y) \)

Quantifier duality: each can be expressed using the other

\( \forall x \ \text{Likes}(x, \text{IceCream}) \)

\( \exists x \ \text{Likes}(x, \text{Broccoli}) \)

Translating environmental information into a FOL
\[ \forall x \ \forall y \text{ is the same as } \forall y \ \forall x \]
\[ \exists x \ \exists y \text{ is the same as } \exists y \ \exists x \]
\[ \exists x \ \forall y \text{ is not the same as } \forall y \ \exists x \]
\[ \exists x \ \forall y \text{ Loves}(x,y) \]
“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

\[ \forall x \ \text{Likes}(x,\text{IceCream}) \quad \neg \exists x \ \neg \text{Likes}(x,\text{IceCream}) \]
\[ \exists x \ \text{Likes}(x,\text{Broccoli}) \quad \neg \forall x \ \neg \text{Likes}(x,\text{Broccoli}) \]

Translating environmental information into a FOL
One with this arrangement:

Two variations with each of these two arrangements:

Four variations with each of these two arrangements:
Block World Syntax

What relations are required

Example:

Describe this situation?
Block World Syntax

In the table below, $x, y \in \{B1, B2, P1, P2\}$.

<table>
<thead>
<tr>
<th>Proposition Schema</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>On_table($x$)</td>
<td>Object $x$ is on the table.</td>
</tr>
<tr>
<td>On($x,y$)</td>
<td>Object $x$ is atop object $y$.</td>
</tr>
<tr>
<td>Is_cube($x$)</td>
<td>Object $x$ is a cube.</td>
</tr>
<tr>
<td>Is_pyramid($x$)</td>
<td>Object $x$ is a pyramid.</td>
</tr>
</tbody>
</table>

Example:

![Diagram of blocks and pyramid]

Describe this situation?
We can Also Make Statements About a Range of Situations

Example: “There is a stack of three objects.”

Example: “There are at least two objects on the table.” This is a bit trickier. A first attempt is as follows:
We can Also Make Statements About a Range of Situations

Example: “There is a stack of three objects.”

\[(\exists x)(\exists y)(\exists z)(\text{On}(y,x) \land \text{On}(z,y))\]

Example: “There are at least two objects on the table.” This is a bit trickier. A first attempt is as follows:

\[(\exists x)(\exists y)(\text{On\_table}(x) \land \text{On\_table}(y))\]

What else is required
Lets Describe Block World Completely (not a situation) - 1
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:

Nothing is both a block and a pyramid:

Domain closure; the only objects are those which are identified explicitly:

Objects are distinct:

No object can rest atop a pyramid.

No object can rest atop another object and lie on the table at the same time.
Every object is either on the table or else atop another object.
Lets Describe Block World Completely (not a situation) - 2

Every object is either on the table or else atop another object.

No object can rest atop itself.

An object can rest atop at most one object.

At most one object can rest atop another object.
Let's Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:

Nothing is both a block and a pyramid:

Domain closure; the only objects are those which are identified explicitly:

Objects are distinct:

No object can rest atop a pyramid.

No object can rest atop another object and lie on the table at the same time.
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
$(\forall x)(\text{Is\_cube}(x) \lor \text{Is\_pyramid}(x))$

Nothing is both a block and a pyramid:

Domain closure; the only objects are those which are identified explicitly:

Objects are distinct:

No object can rest atop a pyramid.

No object can rest atop another object and lie on the table at the same time.
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
\((\forall x)(\text{is\_cube}(x) \lor \text{is\_pyramid}(x)))\)

Nothing is both a block and a pyramid:
\((\forall x)(\neg(\text{is\_cube}(x) \land \text{is\_pyramid}(x))))\)

Domain closure; the only objects are those which are identified explicitly:


Objects are distinct:


No object can rest atop a pyramid.


No object can rest atop another object and lie on the table at the same time.
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
\((\forall x)(\text{is\_cube}(x) \lor \text{is\_pyramid}(x))\)

Nothing is both a block and a pyramid:
\((\forall x)(\neg(\text{is\_cube}(x) \land \text{is\_pyramid}(x)))\)

Domain closure; the only objects are those which are identified explicitly:
\((\forall x)(\text{is\_cube}(x) \leftrightarrow (x=B1 \lor x=B2))\)
\((\forall x)(\text{is\_pyramid}(x) \leftrightarrow (x=P1 \lor x=P2))\)

Objects are distinct:

No object can rest atop a pyramid.

No object can rest atop another object and lie on the table at the same time.
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
(∀x)( ls_cube(x) ∨ ls_pyramid(x))

Nothing is both a block and a pyramid:
(∀x)(¬(ls_cube(x) ∧ ls_pyramid(x)))

Domain closure; the only objects are those which are identified explicitly:
(∀x)( ls_cube(x) ↔ ( x=B1 ∨ x=B2))
(∀x)( ls_pyramid(x) ↔ ( x=P1 ∨ x=P2))

Objects are distinct:
(B1 ≠ B2) ∧ (P1 ≠ P2) ∧ (B1 ≠ P1) ∧ (B1 ≠ P2)
    ∧ (B2 ≠ P1) ∧ (B2 ≠ P2)
(Only the first two statements are necessary, strictly speaking, since the others are deducible.)

No object can rest atop a pyramid.

No object can rest atop another object and lie on the table at the same time.
Lets Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
(∀x)( is_cube(x) ∨ is_pyramid(x))

Nothing is both a block and a pyramid:
(∀x)(¬(is_cube(x) ∧ is_pyramid(x)))

Domain closure; the only objects are those which are identified explicitly:
(∀x)( is_cube(x) ↔ ( x=B1 ∨ x=B2))
(∀x)( is_pyramid(x) ↔ ( x=P1 ∨ x=P2))

Objects are distinct:
(B1 ≠ B2) ∧ (P1 ≠ P2) ∧ (B1 ≠ P1) ∧ (B1 ≠ P2)
       ∧ (B2 ≠ P1) ∧ (B2 ≠ P2)
(Only the first two statements are necessary, strictly speaking, since the others are deducible.)

No object can rest atop a pyramid.
(∀x)(∀y)(¬(is_pyramid(x) ∧ On(y,x)))

No object can rest atop another object and lie on the table at the same time.
Let's Describe Block World Completely (not a situation) - 1

Everything is either a block or a pyramid:
\((\forall x)(\text{is\_cube}(x) \lor \text{is\_pyramid}(x))\)

Nothing is both a block and a pyramid:
\((\forall x)(\neg(\text{is\_cube}(x) \land \text{is\_pyramid}(x)))\)

Domain closure; the only objects are those which are identified explicitly:
\((\forall x)(\text{is\_cube}(x) \leftrightarrow (x=B1 \lor x=B2))\)
\((\forall x)(\text{is\_pyramid}(x) \leftrightarrow (x=P1 \lor x=P2))\)

Objects are distinct:
\((B1 \neq B2) \land (P1 \neq P2) \land (B1 \neq P1) \land (B1 \neq P2)\)
\(\land (B2 \neq P1) \land (B2 \neq P2)\)

(Only the first two statements are necessary, strictly speaking, since the others are deducible.)

No object can rest atop a pyramid.
\((\forall x)(\forall y)(\neg(\text{is\_pyramid}(x) \land \text{on}(y,x)))\)

No object can rest atop another object and lie on the table at the same time.
\((\forall x)(\forall y)(\neg(\text{on\_table}(x) \land \text{on}(x,y)))\)
Let's Describe Block World Completely (not a situation) - 2

Every object is either on the table or else atop another object.

No object can rest atop itself.

An object can rest atop at most one object.

At most one object can rest atop another object.
Every object is either on the table or else atop another object.
$(\forall x)(\exists y)(\text{On}\_\text{table}(x) \lor \text{On}(x,y))$

No object can rest atop itself.

An object can rest atop at most one object.

At most one object can rest atop another object.
Let's Describe Block World Completely (not a situation) - 2

Every object is either on the table or else atop another object.
\((\forall x)(\exists y)(\text{On}\_\text{table}\(x\) \lor \text{On}(x,y)))\)

No object can rest atop itself.
\((\forall x)(\neg \text{On}(x,x)))\)

An object can rest atop at most one object.

At most one object can rest atop another object.
Every object is either on the table or else atop another object.
\((\forall x)(\exists y)(\text{On
d_table}(x) \lor \text{On}(x,y)))\)

No object can rest atop itself.
\((\forall x)(\neg \text{On}(x,x)))\)

An object can rest atop at most one object.
\((\forall x)(\forall y)(\forall z) ((\text{On}(x,y) \land \text{On}(x,z)) \rightarrow y=z))\)

At most one object can rest atop another object.
Lets Describe Block World Completely (not a situation) - 2

Every object is either on the table or else atop another object.
$(\forall x)(\exists y) (\text{On\_table}(x) \lor \text{On}(x,y))$

No object can rest atop itself.
$(\forall x)(\neg \text{On}(x,x))$

An object can rest atop at most one object.
$(\forall x)(\forall y)(\forall z) ((\text{On}(x,y) \land \text{On}(x,z)) \rightarrow y=z)$

At most one object can rest atop another object.
$(\forall x)(\forall y)(\forall z) ((\text{On}(y,x) \land \text{On}(z,x)) \rightarrow y=z)$
Inference in F.O.L

• How?
  – Reduce to propositional logic and then do inference
  – Forwards, backwards chaining
  – Resolution
Lecture After Break

Frame Problem?
The Frame Problem

“To most AI researchers, the frame problem is the challenge of representing the effects of action in logic without having to represent explicitly a large number of intuitively obvious non-effects. To many philosophers, the AI researchers' frame problem is suggestive of a wider epistemological issue, namely whether it is possible, in principle, to limit the scope of the reasoning required to derive the consequences of an action.”
Example of Frame Problem

Suppose we write two formulae, one describing the effects of painting an object and the other describing the effects of moving an object.

1. Colour\((x,c)\) holds after Paint\((x,c)\)
2. Position\((x,p)\) holds after Move\((x,p)\)

Now, suppose we have an initial situation in which Colour\((A,Red)\) and Position\((A,House)\) hold. According to the machinery of deductive logic, what then holds after the action Paint\((A,Blue)\) followed by the action Move\((A,Garden)\)? Intuitively, we would expect Colour\((A,Blue)\) and Position\((A,Garden)\) to hold.

Home work question #1: a) What predicate does hold?, b) why?
Propositionalization

What do we need to get rid of?
Propositionalization

What do we need to get rid of?

**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., $\forall x \text{ King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x)$ yields

- $\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John})$
- $\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard})$
- $\text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \Rightarrow \text{Evil}(\text{Father}(\text{John}))$

;
Existential instantiation (EI)

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

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

E.g., \( \exists x \; \text{Crown}(x) \land \text{OnHead}(x, \text{John}) \) yields
\[
\text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John})
\]

provided \( C_1 \) is a new constant symbol, called a ????????

Another example: from \( \exists x \; d(x^y)/dy = x^y \) we obtain
\[
d(e^y)/dy = e^y
\]

provided \( e \) is a new constant symbol
Existential instantiation (EI)

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

$$\exists v \quad \alpha$$

$$\operatorname{Subst}\{\{v/k\}, \alpha\}$$

E.g., $\exists x \quad \text{Crown}(x) \land \text{OnHead}(x, \text{John})$ yields

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

provided $C_1$ is a new constant symbol, called a Skolem constant

Another example: from $\exists x \quad d(x^y)/dy = x^y$ we obtain

$$d(e^y)/dy = e^y$$

provided $e$ is a new constant symbol
Existential instantiation contd.

UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old.

EI can be applied once to *replace* the existential sentence; the new KB is *not* equivalent to the old, but is satisfiable iff the old KB was satisfiable.
Reduction to propositional inference

Suppose the KB contains just the following:

\[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \]
\[ King(John) \]
\[ Greedy(John) \]
\[ Brother(Richard, John) \]

Instantiating the universal sentence in \textit{all possible} ways, we have
Reduction to propositional inference

Suppose the KB contains just the following:

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

King(John)
Greedy(John)
Brother(Richard, John)

Instantiating the universal sentence in all possible ways, we have

King(John) \land Greedy(John) \Rightarrow Evil(John)
King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)
King(John)
Greedy(John)
Brother(Richard, John)

The new KB is propositionalized: proposition symbols are

King(John), Greedy(John), Evil(John), King(Richard) etc.
Stefan’s Comment/Question

If entailment in propositional logic is decidable and we can reduce F.O.L. to propositional logic, how is entailment in F.O.L. undecidable???

Reduction contd.

Claim: a ground sentence* is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve entailment

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

But we can have an infinite number of ground terms How?

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

Idea: For $n = 0$ to go do
Stefan’s Comment/Question

If entailment in propositional logic is decidable and we can reduce F.O.L. to propositional logic, how is entailment in F.O.L. undecidable???

Reduction contd.

Claim: a ground sentence* is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve entailment

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

Problem: with function symbols, there are infinitely many ground terms, e.g., \( \text{Father}(\text{Father}(\text{Father}(\text{John}))) \)

Theorem: Herbrand (1930). If a sentence \( \alpha \) is entailed by an FOL KB, it is entailed by a finite subset of the propositional 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
Stefan’s Comment/Question

**Reduction contd.**

Claim: a ground sentence* is entailed by new KB iff entailed by original KB

Claim: every FOL KB can be propositionalized so as to preserve entailment

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

Problem: with function symbols, there are infinitely many ground terms, e.g., \( \text{Father(}\text{Father(}\text{Father(John)})) \)

Theorem: Herbrand (1930). If a sentence \( \alpha \) is entailed by an FOL KB, it is entailed by a finite subset of the propositional 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 in FOL is semidecidable
Upto 1960's FOL Inference …

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant sentences. E.g., from

\[
\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x) \\
King(John) \\
\forall y \ Greedy(y) \\
Brother(Richard, John)
\]

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

With \(p\) \(k\)-ary predicates and \(n\) constants, there are \(p \cdot n^k\) instantiations!
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 contd.

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

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

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

Missiles are weapons:

An enemy of America counts as “hostile”:

West, who is American ...

The country Nono, an enemy of America ...
... 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., \( \exists x \text{ Owns}(Nono, x) \land \text{Missile}(x) \):

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

Missiles are weapons:

An enemy of America counts as “hostile”:

West, who is American ...

The country Nono, an enemy of America ...
Example knowledge base contd.

... 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

Missiles are weapons:

An enemy of America counts as “hostile”:

West, who is American ...

The country Nono, an enemy of America ...
Example knowledge base contd.

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

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

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

Missiles are weapons:

An enemy of America counts as “hostile”:

West, who is American ...

The country Nono, an enemy of America ...
Example knowledge base contd.

... 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”:

West, who is American ... 

The country Nono, an enemy of America ...
Example knowledge base contd.

... 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 ...

The country Nono, an enemy of America ...
Example knowledge base contd.

... 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 ...
Example knowledge base contd.

... 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 ...

\[ \text{Enemy}(\text{Nono}, \text{America}) \]
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 \( \text{Greedy}(y) \):

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

**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></td>
</tr>
<tr>
<td>( \text{Knows}(\text{John}, x) )</td>
<td>( \text{Knows}(y, \text{OJ}) )</td>
<td></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., \( \text{Knows}(z_{17}, \text{OJ}) \)

These are simple examples, gets quite involved:

UNIFY(\( \text{Knows}(\text{John}, x) \), \( \text{Knows}(y, z) \))?
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

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

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

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, OJ)$
Generalized Modus Ponens (GMP)

\[
\frac{p_1', \, p_2', \, \ldots, \, p_n', \, (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{q\theta}
\]

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

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

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

Going back to PL. Modus Ponens is a specialization of what?
Forward chaining proof

Example knowledge base contd.

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

\text{Nono} \ldots \text{has some missiles, i.e., } \exists x \ \text{Owns(Nono, x) \land 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":
\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]

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

The country \text{Nono}, an enemy of America ... 
\[ \text{Enemy}(\text{Nono, America}) \]
Forward chaining proof

... 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 proof

Criminal(West)

Weapon(M1)

Sells(West,M1,Nono)

Hostile(Nono)

Enemy(Nono,America)

Owns(Nono,M1)

Missile(M1)

American(West)
What is forward chaining only applicable to?
Properties of forward chaining

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 When? HW Question #2

This is unavoidable: entailment with definite clauses is semidecidable

Used extensively in deductive databases
Efficiency of forward chaining

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
**Backward chaining algorithm**

function FOL-BC-Ask(\( KB, \) goals, \( \theta \)) returns a set of substitutions

inputs: \( KB, \) a knowledge base
   \( goals, \) a list of conjuncts forming a query
   \( \theta, \) the current substitution, initially the empty substitution \( \{ \} \)

local variables: \( ans, \) a set of substitutions, initially empty

if \( goals \) is empty then return \( \{ \theta \} \)

\( q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals)) \)

for each \( r \) in \( KB \) where \( \text{STANDARDIZE-APART}(r) = (p_1 \land \ldots \land p_n \Rightarrow q) \)
   and \( \theta' \leftarrow \text{UNIFY}(q, q') \) succeeds
   \( ans \leftarrow \text{FOL-BC-Ask}(KB, [p_1, \ldots, p_n | \text{REST}(goals)], \text{COMPOSE}(\theta', \theta)) \cup ans \)

return \( ans \)
Backward chaining example

**Example knowledge base contd.**

... 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 example

Example knowledge base contd.

... 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 example

\[
\text{Criminal(West)} \quad \{x/West, y/M1\}
\]

\[
\text{American(West)} \quad \{\}
\]

\[
\text{Weapon(y)} \quad \{\}
\]

\[
\text{Sells(x,y,z)}
\]

\[
\text{Hostile(z)}
\]

\[
\text{Missile(y)} \quad \{y/M1\}
\]

---

Example knowledge base contd.

... 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., \( \exists x \, \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[
\text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1)
\]

... 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 ...
\[
\text{Enemy}(\text{Nono, America})
\]
Backward chaining example

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

\[ \{x/\text{West}, y/\text{M1}, z/\text{Nono}\} \]

- \text{American} \text{(West)}
  \[ \{\} \]
- \text{Weapon} \text{(y)}
- \text{Sells} \text{(West, M1, z)}
  \[ \{z/\text{Nono}\} \]
- \text{Hostile} \text{(z)}
- \text{Missile} \text{(y)}
  \[ \{y/\text{M1}\} \]
- \text{Missile} \text{(M1)}
- \text{Owns} \text{(Nono, M1)}

---

Example knowledge base contd.

- It is a crime for an American to sell weapons to hostile nations:
  \[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
- \text{Nono} \ldots has some missiles, i.e., \( \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
  \[ \text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1) \]
- \ldots 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 \ldots
  \[ \text{American}(\text{West}) \]
- The country \text{Nono}, an enemy of America \ldots
  \[ \text{Enemy}(\text{Nono, America}) \]
Backward chaining example

\[ \text{Criminal(West)} \]
\[ \{x/\text{West}, y/M1, z/\text{Nono}\} \]

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

\[ \text{Weapon}(y) \]
\[ \{ \} \]

\[ \text{Sells(West,M1,z)} \]
\[ \{ z/\text{Nono} \} \]

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

\[ \text{Missile}(y) \]
\[ \{ y/M1 \} \]

\[ \text{Missile(M1)} \]
\[ \{ \} \]

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

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

Example knowledge base contd.

... 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., \( \exists x \text{ Owns}(\text{Nono}, x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \]

... 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 ...
\[ \text{American(West)} \]

The country Nono, an enemy of America ...
\[ \text{Enemy}(\text{Nono, America}) \]
function FOL-BC-Ask(KB, goals, \( \theta \)) returns a set of substitutions

inputs:  
- \( KB \), a knowledge base
- \( goals \), a list of conjuncts forming a query
- \( \theta \), the current substitution, initially the empty substitution \( \{ \} \)

local variables:  
- \( ans \), a set of substitutions, initially empty

if \( goals \) is empty then return \( \{ \theta \} \)

\( q' \leftarrow \text{SUBST}(\theta, \text{FIRST}(goals)) \)

for each \( r \) in \( KB \) where \( \text{STANDARDIZE-APART}(r) = (\ p_1 \land \ldots \land p_n \Rightarrow q) \)
and \( \theta' \leftarrow \text{UNIFY}(q, q') \) succeeds

\( ans \leftarrow \text{FOL-BC-Ask}(KB, [p_1, \ldots, p_n|\text{REST}(goals)], \text{COMPOSE}(\theta', \theta)) \cup ans \)

return \( ans \)
Properties of backward chaining

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
How can we Use FOL to Reason About an Environment

Performance measure

gold +1000, death -1000
-1 per step, -10 for using the arrow

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 Breeze, Glitter, Smell

Actuators Left turn, Right turn,
Forward, Grab, Release, Shoot
Interacting with FOL KB

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

\[
\text{TELL}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5)) \quad \text{Ask}(KB, \exists a \ \text{Action}(a, 5)) \quad \text{For prop logic??？}
\]

i.e., does the KB entail any particular actions at \( t = 5 \)?

Answer: Yes, \( \{a/\text{Shoot}\} \quad \leftarrow \text{substitution} \) (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 \( KB \models S\sigma \)
Perception and Reflex Agents

Reflexes are Useful for Directly Figuring Out What we don’t Know

"Perception"

$\forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smelt}(t)$
$\forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t)$

Reflex: $\forall t \ AtGold(t) \Rightarrow \text{Action}(\text{Grab}, t)$

Reflex with internal state: do we have the gold already?

$\text{Holding}(\text{Gold}, t)$ cannot be observed
$\Rightarrow$ keeping track of change is essential
Perception and Reflex Agents

"Perception"
\[ \forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smelt}(t) \]
\[ \forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t) \]

Reflex: \[ \forall t \ \text{AtGold}(t) \Rightarrow \text{Action(Grab, t)} \]

Reflex with internal state: do we have the gold already?
\[ \forall t \ \text{AtGold}(t) \land \neg \text{Holding(Gold, t)} \Rightarrow \text{Action(Grab, t)} \]

\[ \text{Holding(Gold, t)} \text{ cannot be observed} \]
\[ \Rightarrow \text{keeping track of change is essential} \]

An exercise???
Hidden Property Deduction

Properties of locations:
\( \forall l, t \quad \text{At}(\text{Agent}, l, t) \land \text{Smelt}(t) \Rightarrow \text{Smelly}(l) \)
\( \forall l, t \quad \text{At}(\text{Agent}, l, t) \land \text{Breeze}(t) \Rightarrow \text{Breezy}(l) \)

Squares are breezy near a pit: Can Encode This Two Ways

Diagnostic rule—infer cause from effect

Causal rule—infer effect from cause

Neither of these is complete—e.g., the causal rule doesn’t say whether squares far away from pits can be breezy

Definition for the Breezy predicate:
Hidden Property Deduction

Properties of locations:
\[ \forall l, t \; At(Agent, l, t) \land Smelt(t) \Rightarrow Smelly(l) \]
\[ \forall l, t \; At(Agent, l, t) \land Breeze(t) \Rightarrow Breezy(l) \]

Squares are breezy near a pit:  
Can Encode This Two Ways

**Diagnostic rule—infer cause from effect**
\[ \forall y \; Breezy(y) \Rightarrow \exists x \; Pit(x) \land Adjacent(x, y) \]

**Causal rule—infer effect from cause**
\[ \forall x, y \; Pit(x) \land Adjacent(x, y) \Rightarrow Breezy(y) \]

Neither of these is complete—e.g., the causal rule doesn’t say whether squares far away from pits can be breezy

**Definition for the Breezy predicate:**
\[ \forall y \; Breezy(y) \Leftrightarrow [\exists x \; Pit(x) \land Adjacent(x, y)] \]
Keeping Track of Situations

Facts hold in situations, rather than eternally
E.g., \(\text{Holding}(\text{Gold}, \text{Now})\) rather than just \(\text{Holding}(\text{Gold})\)

Situation calculus is one way to represent change in FOL:
   Adds a situation argument to each non-eternal predicate
E.g., \(\text{Now} \) in \(\text{Holding}(\text{Gold}, \text{Now})\) denotes a situation

Situations are connected by the Result function
\(\text{Result}(a, s)\) is the situation that results from doing \(a\) is \(s\)
Planning

Initial condition in KB:

\[ \text{At}(\text{Agent}, [1, 1], S_0) \]
\[ \text{At}(\text{Gold}, [1, 2], S_0) \]

Query: \text{Ask}(KB, \exists s \ \text{Holding}(\text{Gold}, s))

i.e., in what situation will I be holding the gold?

Answer: \{s/\text{Result}(\text{Grab}, \text{Result}(\text{Forward}, S_0))\}

i.e., go forward and then grab the gold

This assumes that the agent is interested in plans starting at \( S_0 \) and that \( S_0 \) is the only situation described in the KB

So as we go along, add in states we observe (like Q-learning)
Beginning To Plan in BlockWorld

• HW Question #3.
• Change (if necessary) the basic sentences in block-world to represent a situation calculus so that the following two questions can be asked:
  – Does B1 sit on B2?
  – What does B1 sit on?
  – Is P1 underneath B2?
Better Planning

Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\)

\(\text{PlanResult}(p, s)\) is the result of executing \(p\) in \(s\)

Then the query \(\text{ASK}(KB, \exists p \ \text{Holding}(\text{Gold}, \text{PlanResult}(p, S_0)))\)
has the solution \(\{p/\{\text{Forward, Grab}\}\}\)

Definition of \(\text{PlanResult}\) in terms of \(\text{Result}\):
\[
\forall s \ \text{PlanResult}([], s) = s \\
\forall a, p, s \ \text{PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))
\]

Planning systems are special-purpose reasoners designed to do this type
of inference more efficiently than a general-purpose reasoner.