Knowledge-based Agent

- Knowledge base
  - \texttt{TELL} agent about the environment

- Knowledge representation
  - Propositional logic
  - First-order logic
  - Many others...

- Reasoning via inference
  - \texttt{ASK} agent how to achieve goal based on current knowledge
function **KB-Agent**(*percept*) returns an action

**persistent**: *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*
Goals
- Visit safe locations
- Grab gold if present
- If have gold or no more safe, unvisited locations, then move to $[1,1]$ and Climb
Acting Logically in Wumpus World
Acting Logically in Wumpus World

Percept₁ = [None,None,None,None,None]
- [1,2] and [2,1] safe
Action = GoForward
Percept₂ = [None,Breeze,None,None,None]
- Either [2,2] or [3,1] or both has a pit
Execute TurnLeft, TurnLeft, GoForward, TurnRight, GoForward
Acting Logically in Wumpus World

- Percept_7 = [Stench,None,None,None,None,None]
  - Wumpus in [1,3]
  - No pit in [2,2] (safe), so pit in [3,1]
- Could Shoot, but <TurnRight,GoForward> to [2,2]
- Percept_9 = [None,None,None,None,None,None]
  - [3,2] and [2,3] are safe
- <TurnLeft,GoForward> to [2,3]
- Percept_11 = [Stench,Breeze,Glitter,None,None,None]
- Grab gold, head home, and Climb (score: 1000 - 17 = 983)
A knowledge base (KB) consists of “sentences”

- **Syntax** specifies a well-formed sentence
- **Semantics** specifies the meaning of a sentence
- A **model** $m$ specifies whether each sentence is true or false
  - There can be many or infinite models
  - E.g., Model $m_1$ may say “wumpus in [2,3]” is true, $m_1(\text{wumpus in [2,3]})$
  - E.g., Model $m_2$ may say “wumpus in [2,3]” is false, or “wumpus not in [2,3]”, $m_2(\text{wumpus not in [2,3]})$
- $m(\alpha)$ says “$m$ **satisfies** $\alpha$” or “$m$ is a **model of** $\alpha$”
Logic

- **Entailment** between sentences implies that one sentence follows logically from another.
- $\alpha \models \beta$ means $\alpha$ entails $\beta$.
  - Or, every model in which $\alpha$ is true, $\beta$ is also true.
  - $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$.
  - $M(\alpha)$ is the set of all models in which $\alpha$ is true.
- E.g., $\text{Pit}([3,1]) \models \text{Breeze}([2,1])$. 
Logic

- E.g., $\text{KB} = \{\neg \text{Breeze}([1,1]), \text{Breeze}([2,1]), \text{rules of wumpus world}\}$
- $\text{Pit}([1,2]) \text{ ?} , \text{Pit}([2,2]) \text{ ?} , \text{Pit}([3,1]) \text{ ?}$
- $\alpha_1 = \neg \text{Pit}([1,2]), \text{KB} \models \alpha_1, \text{M}(\text{KB}) \subseteq \text{M}(\alpha_1)$
E.g., KB = \{¬\text{Breeze}(1,1), \text{Breeze}(2,1), \text{rules of wumpus world}\}

\(\alpha_2 = ¬\text{Pit}(2,2)\), KB \nmodels \(\alpha_2\), M(KB) \nmodels M(\alpha_2)
Logical inference is the process by which we infer one sentence is true from others.

E.g., model checking (previous two slides)
- Enumerate all possible models in which KB is true and check if $\alpha$ is also true.

$\text{KB} \vdash_{i} \alpha$ means that $\alpha$ can be derived from KB using inference algorithm $i$.

Inference algorithm $i$ is **sound** or truth-preserving if everything derived is also entailed.

Inference algorithm $i$ is **complete** if it can derive everything that is entailed.
Logic

- Grounding is the connection between logic and the real environment
Syntax

- Atomic sentences consist of a single propositional symbol, which can be true or false.

\[
\begin{align*}
\text{Sentence} & \rightarrow \text{AtomicSentence} \mid \text{ComplexSentence} \\
\text{AtomicSentence} & \rightarrow \text{True} \mid \text{False} \mid P \mid Q \mid R \mid \text{North} \mid W_{1,3} \mid \ldots \\
\text{ComplexSentence} & \rightarrow (\text{Sentence}) \mid [\text{Sentence}] \\
\phantom{\text{ComplexSentence}} & \mid \neg \text{Sentence} \quad \text{“not”} \\
\phantom{\text{ComplexSentence}} & \mid \text{Sentence} \land \text{Sentence} \quad \text{“and”} \\
\phantom{\text{ComplexSentence}} & \mid \text{Sentence} \lor \text{Sentence} \quad \text{“or”} \\
\phantom{\text{ComplexSentence}} & \mid \text{Sentence} \Rightarrow \text{Sentence} \quad \text{“implies”} \\
\phantom{\text{ComplexSentence}} & \mid \text{Sentence} \Leftrightarrow \text{Sentence} \quad \text{“if and only if”}
\end{align*}
\]

Operator Precedence: \( \neg, \land, \lor, \Rightarrow, \Leftrightarrow \)
Propositional Logic

- **Syntax**
  - Not (¬) is a negation
  - **Literal** is either an atomic sentence (positive literal) or a negated atomic sentence (negative literal)
    - ¬B₁,₁, B₂,₁
  - And (∧) is a conjunction; its parts are **conjuncts**
  - Or (∨) is a disjunction; its parts are **disjuncts**
  - Implies (⇒) is an **implication** (P₂,₂ ⇒ B₁,₂)
    - Its lefthand side is the **antecedent** or **premise**
    - Its righthand side is the **consequent** or **conclusion**
  - If and only if (⇔) is a **biconditional**
Propositional Logic

Semantics
- How to determine the truth value (true or false) of every proposition in a model
- True is true in every model
- False is false in every model
- Truth values of every other proposition must be specified directly in the model
  - E.g., \( m_1 = \{W_{1,3}=\text{true}, \ P_{3,1}=\text{true}, \ P_{2,2}=\text{false}, \ldots\} \)
Propositional Logic

- Semantics for complex sentences in model m
  - $\neg P$ is true iff $P$ is false in m
  - $P \land Q$ is true iff both $P$ and $Q$ are true in m
  - $P \lor Q$ is true iff either $P$ or $Q$ is true in m
  - $P \Rightarrow Q$ is true unless $P$ is true and $Q$ is false in m
  - $P \Leftrightarrow Q$ is true iff $P$ and $Q$ are both true or both false in m

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>$\neg P$</th>
<th>$P \land Q$</th>
<th>$P \lor Q$</th>
<th>$P \Rightarrow Q$</th>
<th>$P \Leftrightarrow Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</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>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Propositional Wumpus World KB

- \( P_{x,y} \) is true if there is a pit in \([x,y]\)
- \( W_{x,y} \) is true if there is a wumpus in \([x,y]\), alive or dead
- \( B_{x,y} \) is true if the agent perceives a breeze in \([x,y]\)
- \( S_{x,y} \) is true if the agent perceives a stench in \([x,y]\)

\[\begin{array}{cccc}
1,4 & 2,4 & 3,4 & 4,4 \\
1,3 & 2,3 & 3,3 & 4,3 \\
1,2 & 2,2 & 3,2 & 4,2 \\
1,1 & 2,1 & 3,1 & 4,1 \\
\end{array}\]

- \( R_1: \neg P_{1,1} \)
- \( R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \)
- \( R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \)
- \( R_4: \neg B_{1,1} \)
- \( R_5: B_{2,1} \)
Simple Propositional Inference

function \( \text{TT-ENTAILS} \)\( (KB, \alpha) \) returns true or false \( // KB \models \alpha \)?

\( \text{symbols} \leftarrow \) a list of the proposition symbols in \( KB \) and \( \alpha \)
return \( \text{TT-CHECK-ALL} \)\( (KB, \alpha, \text{symbols}, \{\}) \)

function \( \text{TT-CHECK-ALL} \)\( (KB, \alpha, \text{symbols}, \text{model}) \) returns true or false

if \( \text{EMPTY?} \)\( (\text{symbols}) \) then
  if \( \text{PL-TRUE?} \)\( (KB, \text{model}) \) then return \( \text{PL-TRUE?} \)\( (\alpha, \text{model}) \)
  else return \( \text{true} \) \( \text{if KB false, return true?} \)
else do
  \( P \leftarrow \text{FIRST} \)\( (\text{symbols}) \)
  \( \text{rest} \leftarrow \text{REST} \)\( (\text{symbols}) \)
return \( \text{TT-CHECK-ALL} \)\( (KB, \alpha, \text{rest, model} \cup \{P = \text{true}\}) \) \( \text{and} \)
  \( \text{TT-CHECK-ALL} \)\( (KB, \alpha, \text{rest}, \text{model} \cup \{P = \text{false}\}) \)) \( \text{and?} \)

\( \text{PL-TRUE?} \)\( (s, m) \) returns true if sentence \( s \) true in model \( m \).
\( \text{TT-ENTAILS?} \) sound and complete, but \( O(2^n) \) time complexity.
What Happened to HAL?

“2001: A Space Odyssey” (1968)
Propositional Theorem Proving

- Theorem proving
  - Applying rules of inference on sentences in KB to derive sentence $\alpha$
- A sentence is valid if it is true in all models
- Deduction theorem
  - For any sentences $\alpha$ and $\beta$, $\alpha \models \beta$ if and only if the sentence ($\alpha \Rightarrow \beta$) is valid
A sentence is satisfiable if it is true in some model
- $\alpha \models \beta$ if and only if the sentence $(\alpha \land \neg \beta)$ is unsatisfiable

Proof by refutation or contradiction
- Prove $\beta$ from $\alpha$ by checking if $(\alpha \land \neg \beta)$ is unsatisfiable
- I.e., assume $\beta$ to be false and show this leads to a contradiction with $\alpha$
- Also implies that an inconsistent $\alpha$ can prove anything
What Really Happened to HAL?

Logical Equivalences

\[(\alpha \land \beta) \equiv (\beta \land \alpha)\]  commutativity of \(\land\)

\[(\alpha \lor \beta) \equiv (\beta \lor \alpha)\]  commutativity of \(\lor\)

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

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

\[-(-\alpha) \equiv \alpha\]  double-negation elimination

\[(\alpha \implies \beta) \equiv (-\beta \implies -\alpha)\]  contraposition

\[(\alpha \implies \beta) \equiv (-\alpha \lor \beta)\]  implication elimination

\[(\alpha \iff \beta) \equiv ((\alpha \implies \beta) \land (\beta \implies \alpha))\]  biconditional elimination

\[-(\alpha \land \beta) \equiv (-\alpha \lor -\beta)\]  de Morgan

\[-(\alpha \lor \beta) \equiv (-\alpha \land -\beta)\]  de Morgan

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

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

- **Notation:**
  \[ \begin{array}{c}
  \text{sentences given} \\
  \text{sentences inferred}
  \end{array} \]

- **Modus Ponens:**
  \[
  \alpha \Rightarrow \beta, \quad \alpha \quad \Rightarrow \beta
  \]

- **And–Elimination:**
  \[
  \alpha \land \beta \\
  \Rightarrow \alpha
  \]

- **Biconditional elimination:**
  \[
  \alpha \Leftrightarrow \beta \\
  (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)
  \]
Sample Proof

$R_1$: $\neg P_{1,1}$
$R_2$: $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
$R_3$: $B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
$R_4$: $\neg B_{1,1}$
$R_5$: $B_{2,1}$

Prove: $\neg P_{1,2}$

Apply biconditional elimination to $R_2$
$R_6$: $(B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

Apply And-Elimination to $R_6$
$R_7$: $((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

Apply Contraposition to $R_7$
$R_8$: $(\neg B_{1,1} \Rightarrow \neg(P_{1,2} \lor P_{2,1}))$

Apply Modus Ponens to $R_8$ and $R_4$
$R_9$: $\neg(P_{1,2} \lor P_{2,1})$

Apply De Morgan’s rule to $R_9$
$R_{10}$: $\neg P_{1,2} \land \neg P_{2,1}$

Apply And-Elimination to $R_{10}$
$\neg P_{1,2}$ (done)
Proof by Search

- Use search to perform propositional inference
- **Initial State**: KB
- **Actions**: Apply inference rules to sentences matching top of rule
- **Result**: Add sentences in bottom of rule to KB
- **Goal**: KB contains sentence to be proved
- Sound?
- Complete?
- Efficient?
Resolution

- \{(A \Rightarrow B), A\} \models B \text{ "modus ponens"}
- \{(-A \lor B), A\} \models B \text{ "unit resolution"}

- \((A \land B \land C) \Rightarrow D\)
  - \neg(A \land B \land C) \lor D
  - \neg A \lor \neg B \lor \neg C \lor D

- Full resolution
  - \{(-A \lor \neg B \lor \neg C \lor D), (A \lor E)\} \models \neg B \lor \neg C \lor D \lor E

Example
- \((A \Rightarrow C), (B \Rightarrow C), (A \lor B)\)
- \((\neg A \lor C), (\neg B \lor C), (A \lor B)\)
- \((B \lor C), (\neg B \lor C)\)
- \((C \lor C) \models C\)
Example (cont.)

- Agent moves to [1,2]; prove: $P_{3,1}$

Add percept information to KB

- $R_{11}: \neg B_{1,2}$
- $R_{12}: B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3})$

By earlier process, can eliminate pits in [2,2] and [1,3]

- $R_{13}: \neg P_{2,2}$
- $R_{14}: \neg P_{1,3}$

Apply Biconditional Elimination to $R_3$, then Modus Ponens with $R_5$

- $R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1}$

Apply Resolution to $R_{13}$ and $R_{15}$

- $R_{16}: P_{1,1} \lor P_{3,1}$

Apply Resolution to $R_1$ and $R_{16}$

- $R_{17}: P_{3,1}$
Unit Resolution

- Given literals $l_1,\ldots,l_k$ and $m$, where $l_i$ and $m$ are complementary
- Unit resolution inference rule:

\[
\frac{l_1 \lor \cdots \lor l_k, \quad m}{l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k}
\]

- **Clause** is a disjunction of literals
  - $l_1 \lor \cdots \lor l_k$ is a clause
  - $m$ is a unit clause
Given literals \( l_1, \ldots, l_k \) and \( m_1, \ldots, m_n \), where \( l_i \) and \( m_j \) are complementary

Resolution inference rule:

\[
\begin{align*}
l_1 \lor \cdots \lor l_k, & \quad m_1 \lor \cdots \lor m_n \\
l_1 \lor \cdots \lor l_{i-1} \lor l_i+1 \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n
\end{align*}
\]

Resolution is sound and complete!

Requires sentences to be clauses

Luckily, every sentence in propositional logic can be expressed as a conjunction of clauses
Conjunctive Normal Form (CNF)

- Procedure for converting propositional logic sentence to CNF (*clauses*)
  1. Eliminate $\iff$, replacing $\alpha \iff \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
  2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$.
  3. Move $\neg$ inward to appear only in literals
     - $\neg(\neg \alpha) \equiv \alpha$ (double-negation elimination)
     - $\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta)$ (De Morgan)
     - $\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta)$ (De Morgan)
  4. Apply distributivity of $\lor$ over $\land$
     - $(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma))$
CNF Conversion Example

- $W_{1,3} \iff S_{1,4} \land S_{1,2} \land S_{2,3}$
- **Step 1: Eliminate $\iff$**
  - $(W_{1,3} \Rightarrow S_{1,4} \land S_{1,2} \land S_{2,3}) \land (S_{1,4} \land S_{1,2} \land S_{2,3} \Rightarrow W_{1,3})$
- **Step 2: Eliminate $\Rightarrow$**
  - $(\neg W_{1,3} \lor (S_{1,4} \land S_{1,2} \land S_{2,3})) \land (\neg(S_{1,4} \land S_{1,2} \land S_{2,3}) \lor W_{1,3})$
- **Step 3: Move $\neg$ inward**
  - $(\neg W_{1,3} \lor (S_{1,4} \land S_{1,2} \land S_{2,3})) \land (\neg S_{1,4} \lor \neg S_{1,2} \lor \neg S_{2,3} \lor W_{1,3})$
- **Step 4: Apply distributivity of $\lor$ over $\land$**
  - $(\neg W_{1,3} \lor S_{1,4}) \land (\neg W_{1,3} \lor S_{1,2}) \land (\neg W_{1,3} \lor S_{2,3}) \land (\neg S_{1,4} \lor \neg S_{1,2} \lor \neg S_{2,3} \lor W_{1,3})$
Another Resolution Example

- From previous conversion plus And–Elimination:
  - C1: (¬W₁,₃ ∨ S₁,₄)
  - C2: (¬W₁,₃ ∨ S₁,₂)
  - C3: (¬W₁,₃ ∨ S₂,₃)
  - C4: (¬S₁,₄ ∨ ¬S₁,₂ ∨ ¬S₂,₃ ∨ W₁,₃)
- Assume we have observed the surrounding stenches:
  - C5: S₁,₄
  - C6: S₁,₂
  - C7: S₂,₃
  - Resolving C4 with C5:
    - C8: (¬S₁,₂ ∨ ¬S₂,₃ ∨ W₁,₃)
  - Resolving C8 with C6:
    - C9: (¬S₂,₃ ∨ W₁,₃)
  - Resolving C9 with C7:
    - W₁,₃
Propositional Logic Resolution

function \texttt{PL-RESOLUTION?} (\textit{KB}, \alpha) returns true or false

\begin{itemize}
  \item \textit{clauses} \leftarrow \text{convert} (\textit{KB} \land \neg \alpha) \text{ to CNF}
  \item \textit{new} \leftarrow \{\}
  \item \text{loop do}
    \begin{itemize}
      \item \textbf{for each} pair of clauses \textit{C}_i, \textit{C}_j \textbf{in} \textit{clauses} do
        \begin{itemize}
          \item \textit{resolvents} \leftarrow \texttt{PL-RESOLVE} (\textit{C}_i, \textit{C}_j)
          \item \textbf{if} \textit{resolvents} \textbf{contains} the empty clause \textbf{then return} true
          \item \textit{new} \leftarrow \textit{new} \cup \textit{resolvents}
          \item \textbf{if} \textit{new} \subseteq \textit{clauses} \textbf{then return} false
        \end{itemize}
    \end{itemize}
  \item \textit{clauses} \leftarrow \textit{clauses} \cup \textit{new}
\end{itemize}

\begin{itemize}
  \item Proof by contradiction
  \item Empty clause result of resolving \textit{A} with \neg \textit{A}
  \item If iteration produces no new clauses, then \textit{KB} \not\models \alpha
\end{itemize}
Example

- $KB = R_2 \land R_4 = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$
Efficiency of PL–Resolution

- Branching factor $|KB|^2$
- Many heuristics apply
  - E.g., prefer unit clauses
- Local search works surprisingly well
- **Horn clause** is a clause with at most one positive literal
  - E.g., $(A \land B \land C \Rightarrow D) \equiv (\neg A \lor \neg B \lor \neg C \lor D)$
  - Entailment with Horn clauses is linear in $|KB|$
PL-Based Agent

- KB includes initial state and axioms
  - $\neg W_{1,1} \land \neg P_{1,1} \land L_{1,1} \land \text{FacingEast} \land \text{HaveArrow} \land \neg \text{HaveGold} \land \text{AgentAlive} \land \text{InCave} \land \text{WumpusAlive}$
  - $B_{1,1} \iff (P_{1,2} \lor P_{2,1})$
  - $S_{1,1} \iff (W_{1,2} \lor W_{2,1})$
  - ... 
  - At most one wumpus (e.g., $W_{1,2} \Rightarrow \neg W_{1,3}$)
    - $\neg (W_{1,2} \land W_{1,3}), \neg (W_{1,2} \land W_{1,4}), ...$

- Would prefer
  - $B_{x,y} \iff (P_{x-1,y} \lor P_{x+1,y} \lor P_{x,y-1} \lor P_{x,y+1})$
  - First-order logic
PL-Based Agent

- **Movement**
  - E.g., TurnLeft
    - Add \( \neg \text{FacingEast} \) and FacingNorth to KB
    - Remove FacingEast; add FacingNorth
  - E.g., GoForward
    - Remove \( L_{1,1} \); add \( L_{2,1} \)
    - Bump?
  - E.g., Shoot
    - Remove HaveArrow, add \( \neg \text{HaveArrow} \)
    - WumpusAlive?

- **Monotonic logic**
  - Set of entailed sentences can only increase

- **History?** (e.g., \( L_{1,1} \) at \( t=1 \))
PL-Based Agent

- Frame problem
  - Describing what changes and what stays the same after taking an action

- Frame axiom
  - E.g., $\text{HaveArrow}^{t+1} \Leftrightarrow \text{HaveArrow}^t \land \neg \text{Shoot}^t$
  - Requires sets of axioms for each time transition
  - Or, first-order-ness
PL-Based Agent

- Intermediate information
  - OK\(_{2,1}\)
  - \(\neg\)Visited\(_{2,1}\)

- Goals
  - HaveGold \(\land\) \(\neg\)InCave
  - OK\(_{x,y}\) \(\Rightarrow\) Visited\(_{x,y}\)
  - \(\neg\)WumpusAlive

- In general, pure PL-based agent cumbersome
- Need hybrid logic/search-based agent
function HYBRID-WUMPUS-AGENT(\textit{percept}) \textbf{returns} an action
inputs: \textit{percept}, a list, $[\text{stench}, \text{breeze}, \text{glitter}, \text{bump}, \text{scream}]$
persistent: $KB$, a knowledge base, initially the atemporal “wumpus physics” $t$, a counter, initially 0, indicating time $plan$, an action sequence, initially empty

$TELL(KB, \text{MAKE-PERCEPT-SENTENCE}(\textit{percept}, t))$
Tell the $KB$ the temporal “physics” sentences for time $t$
safe $\leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$

\textbf{if} $\text{ASK}(KB, \text{Glitter}^t) = true$ \textbf{then}

$plan \leftarrow \{\text{Grab}\} + \text{PLAN-ROUTE}(\textit{current}, \{[1,1]\}, \text{safe}) + \{\text{Climb}\}$

\textbf{if} $plan$ is empty \textbf{then}

$unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^t_{x,y}) = false \text{ for all } t' \leq t\}$

$plan \leftarrow \text{PLAN-ROUTE}(\textit{current}, unvisited \cap \text{safe}, \text{safe})$

\textbf{if} $plan$ is empty and $\text{ASK}(KB, \text{HaveArrow}^t) = true$ \textbf{then}

$\text{possible_wumpus} \leftarrow \{[x, y] : \text{ASK}(KB, \neg W_{x,y}) = false\}$

$plan \leftarrow \text{PLAN-SHOT}(\textit{current}, \text{possible_wumpus}, \text{safe})$

\textbf{if} $plan$ is empty \textbf{then}  // no choice but to take a risk

$\text{not_unsafe} \leftarrow \{[x, y] : \text{ASK}(KB, \neg OK^t_{x,y}) = false\}$

$plan \leftarrow \text{PLAN-ROUTE}(\textit{current}, unvisited \cap \text{not_unsafe}, \text{safe})$

\textbf{if} $plan$ is empty \textbf{then}

$plan \leftarrow \text{PLAN-ROUTE}(\textit{current}, \{[1,1]\}, \text{safe}) + \{\text{Climb}\}$

$\text{action} \leftarrow \text{POP}(plan)$

$TELL(KB, \text{MAKE-ACTION-SENTENCE}(\text{action}, t))$

$t \leftarrow t + 1$

\textbf{return} $\text{action}$
Hybrid Wumpus Agent (cont.)

function PLAN-ROUTE(current, goals, allowed) returns an action sequence
inputs: current, the agent’s current position
goals, a set of squares; try to plan a route to one of them
allowed, a set of squares that can form part of the route

problem ← ROUTE-PROBLEM(current, goals, allowed)
return A*-GRAPH-SEARCH(problem)
First-Order Logic

- Propositional logic insufficient to express even commonsense knowledge
- First-order logic (FOL) or First-order predicate calculus (FOPC)
- Borrowing from elements of natural language
  - **Objects**: nouns, noun phrases (e.g., wumpus, pit)
  - **Relations**: verbs, verb phrases (e.g., shoot)
    - **Properties**: adjectives (e.g., smelly)
    - **Functions**: map input to single output (e.g., location(wumpus))
Ontological View of Logics

- **Propositional logic** assumes world consists of facts that are either true, false or unknown
  - E.g., \( \text{wumpus}(1,3) \Rightarrow \text{stench}(1,2) \)

- **First-order logic** assumes world consists of facts, objects and relations that are either true, false or unknown
  - E.g., \( \text{wumpus}(X,Y) \Rightarrow \text{stench}(X,Y-1) \)

- **Temporal logic** = FOL where facts hold at particular times
  - E.g., \( \text{before} \left( \text{action} \left( \text{shoot}, \text{percept} \left( \text{scream} \right) \right) \right) \)

- **Higher-order logic** assumes world includes first-order relations as objects
  - E.g., \( \text{know} \left( \left[ \text{wumpus}(X,Y) \Rightarrow \text{stench}(X,Y-1) \right] \right) \)

- **Probabilistic logic** = propositional logic with a degree of belief for each fact
  - E.g., \( P(\text{wumpus}(1,3)) = 0.067 \)


**FOL Syntax**

<table>
<thead>
<tr>
<th>Sentence</th>
<th>AtomicSentence</th>
<th>ComplexSentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtomicSentence</td>
<td>Predicate</td>
<td>Predicate (Term,...)</td>
</tr>
<tr>
<td>ComplexSentence</td>
<td>(Sentence)</td>
<td>[Sentence]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>¬ Sentence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sentence ∧ Sentence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sentence ∨ Sentence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sentence ⇒ Sentence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sentence ⇔ Sentence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quantifier Variable,... Sentence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Function (Term,...)</th>
<th>Constant</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantifier</td>
<td>∀</td>
<td>∃</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>A</td>
<td>B</td>
<td>Wumpus</td>
</tr>
<tr>
<td>Variable</td>
<td>a</td>
<td>x</td>
<td>s</td>
</tr>
<tr>
<td>Predicate</td>
<td>True</td>
<td>False</td>
<td>Adjacent</td>
</tr>
<tr>
<td>Function</td>
<td>Location</td>
<td>RightOf</td>
<td>...</td>
</tr>
</tbody>
</table>

**Operator Precedence:** ¬, =, ∧, ∨, ⇒, ⇔
**FOL Semantics**

- **Constant symbols** stand for objects
- **Predicate symbols** stand for relations
- **Function symbols** stand for functions
- **R&N convention**: Above symbols begin with uppercase letters
  - E.g., *Wumpus, Adjacent, RightOf*
- **Arity** is the number of arguments to a predicate or function
  - E.g., *Adjacent (loc₁, loc₂), RightOf (location)*
FOL Semantics

- Terms represent objects with constants, variables or functions
- Note: Functions do not return an object, but represent that object
  - E.g., \( \text{Action(GoForward,}t) \land \text{Orientation(Agent, Right,} t) \land \text{At(Agent, loc,} t) \Rightarrow \text{At(Agent, RightOf(loc),} t+1) \)
- R&N convention: variables begin with lowercase letters
Quantifiers

- Express properties of collections of objects
- Universal quantification (\(\forall\))
  - A statement is true for all objects represented by quantified variables
  - E.g., \(\forall x,y\) At(Wumpus,\(x,y\)) \(\Rightarrow\) Stench(\(x+1,y\))
  - Same as \(\forall x,y\) At(Wumpus,\(x,y\)) \(\land\) Stench(\(x+1,y\))
  - Same as \(\forall x,y\) \(\neg\)At(Wumpus,\(x,y\)) \(\lor\) Stench(\(x+1,y\))
- \(\forall x\) P(\(x\)) \(\equiv\) P(A) \(\land\) P(B) \(\land\) P(Wumpus) \(\land\) ...
Existential quantification (∃)
  - There exists at least one set of objects, represented by quantified variables, for which a statement is true
  - E.g., ∃ w,x,y At(w,x,y) \land Wumpus(w)
  - Same as ∃ w,x,y At(w,x,y) \implies Wumpus(w) ?
  - ∃x P(x) \equiv P(A) \lor P(B) \lor P(Wumpus) \lor ...
Properties of Quantifiers

- Nested quantifiers
- $\forall x \ \forall y$ same as $\forall y \ \forall x$ same as $\forall x, y$
- $\exists x \ \exists y$ same as $\exists y \ \exists x$ same as $\exists x, y$
- $\exists x \ \forall y$ same as $\forall y \ \exists x$ ?
  - $\exists x \ \forall y \ \text{Likes}(x, y)$ ?
  - $\forall y \ \exists x \ \text{Likes}(x, y)$ ?
  - $\forall x \ \exists y \ \text{Likes}(x, y)$ ?
  - $\exists y \ \forall x \ \text{Likes}(x, y)$ ?
Properties of Quantifiers

- Negation and quantifiers
  - $\exists x \ P(x) \equiv \neg \forall x \ \neg P(x)$
    - “If P is true for some x, then P can’t be false for all x”
  - $\forall x \ P(x) \equiv \neg \exists x \ \neg P(x)$
    - “If P is true for all x, then there can’t be an x for which P is false.”
  - $\forall x \ \neg P(x) \equiv \neg \exists x \ P(x)$
    - “If P is false for all x, then there can’t be an x for which P is true.”
  - $\neg \forall x \ P(x) \equiv \exists x \ \neg P(x)$
    - “If P is not true for all x, then there must be an x for which P is false.”
Equality

- **Equality symbol** \( (\text{Term1} = \text{Term1}) \) means Term1 and Term2 refer to the same object
  - E.g., \( \text{RightOf(Location(1,1))} = \text{Location(2,1)} \)
- Useful for constraining two terms to be different
  - E.g., **Sibling**
    - \( \text{Sibling}(x,y) \iff \text{Parent}(p,x) \land \text{Parent}(p,y) \)
    - \( \text{Sibling}(x,y) \iff \text{Parent}(p,x) \land \text{Parent}(p,y) \land \neg(x = y) \)
    - \( \forall x,y \text{ Sibling}(x,y) \iff \exists p \text{ Parent}(p,x) \land \text{Parent}(p,y) \land \neg(x = y) \)
Closed-World Assumption

- How do we express that there is only one wumpus?
  - $\forall x,y (\text{Wumpus}(x,y) \Rightarrow \neg(\exists w,z \text{ Wumpus}(w,z) \land (\neg(w = x) \lor \neg(z=y))))$

- How about one arrow? One gold? At least one pit?

- Closed-world assumption
  - Atomic sentences not known to be true are assumed false

- Unique-names assumption
  - Every constant symbol refers to a distinct object

- Domain closure
  - If not named by a constant symbol, then doesn’t exist
Using FOL

- Carefully...

Monty Python and the Holy Grail (1975)
Using FOL

- **TELL (KB, \(\alpha\))**
  - **TELL (KB, Percept([st,br,Glitter,bu,sc],5))**

- **ASK (KB, \(\beta\))**
  - **ASK (KB, \(\exists a\) Action(a,5))**
  - I.e., does KB entail any particular actions at time 5?
  - Answer: Yes, \{a/Grab\} \leftarrow substitution (binding list)

- **ASKVARS (KB, \(\alpha\))**
  - Returns answers (variable bindings) that make \(\alpha\) true
  - Or, use Answer literal (later)
  - **ASK (KB, \(\exists a\) Action(a,5) \land Answer(a))**
FOL for the Wumpus World

- **Percepts**
  - Percept(p,t) = predicate that is true if percept p observed at time t
  - Percept is a list of five terms
  - E.g., Percept([Stench, Breeze, Glitter, None, None], 5)

- **Actions**
  - GoForward, TurnLeft, TurnRight, Grab, Shoot, Climb

- **AskVars** (∃a BestAction(a, 5)) → {a/Grab}
FOL for the Wumpus World

“Perception”
- \( \forall t,s,g,m,c \ Percept([s,\text{Breeze},g,m,c],t) \Rightarrow \text{Breeze}(t) \)
- \( \forall t,s,b,m,c \ Percept([s,b,\text{Glitter},m,c],t) \Rightarrow \text{Glitter}(t) \)

Reflex agent
- \( \forall t \ \text{Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab},t) \)
FOL for the Wumpus World

- Location list term \([x,y]\) (e.g., \([1,2]\))
  - \(\text{Pit}(s)\) or \(\text{Pit}([x,y])\)
  - \(\text{At(Wumpus,}[x,y],t)\)
  - \(\text{At(Agent,}[1,1],1)\)

- Definition of Breezy(s), where \(s\) is a location
  - \(\forall s \text{ Breezy}(s) \iff \exists r \text{ Adjacent}(s,r) \land \text{Pit}(r)\)

- Definition of Adjacent
  - \(\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \iff (x=a \land (y=b-1 \lor y=b+1)) \lor (y=b \land (x=a-1 \lor x=a+1))\)
FOL for the Wumpus World

- Movement
- Wumpus never moves
  - $\forall t \: At(Wumpus, [1,3], t)$
- Nothing can be in two places at once
  - $\forall x, s_1, s_2, t \: At(x, s_1, t) \land At(x, s_2, t) \implies s_1 = s_2$

- Successor-state axioms for each action
  - Describes what’s true before and after action
  - $\forall t \: HaveArrow(t+1) \iff (HaveArrow(t) \land \neg Action(Shoot, t))$
  - $\forall t \: HaveGold(t+1) \iff (HaveGold(t) \lor (Glitter(t) \land Action(Grab, t)))$
  - ...

Artificial Intelligence
Now that we have FOL, how can we perform sound, complete and efficient inference?

Approaches
- Convert to propositional logic
- Generalized Modus Ponens
- Forward and backward chaining
- Logic programming (Prolog)
- Resolution

State of the art
Note of Caution: Bomb #20

Dark Star (1974)
Propositionalization

- Convert FOL problem to PL problem
- Main challenge: Remove quantifiers
- Universal instantiation

$$\forall \nu \alpha \quad \text{SUBST}({\nu/g}, \alpha)$$

- Substitute every ground term $g$ for any variable $v$ in $\alpha$
- E.g., $\forall s, t, r \, \text{At}(Wumpus, s, t) \land \text{Adjacent}(s, r) \Rightarrow \text{Stench}(r)$
  - $\text{At}(Wumpus, [1,3], 1) \land \text{Adjacent}([1,3],[2,3]) \Rightarrow \text{Stench}([2,3])$
  - $\text{At}(Wumpus, [2,2], 1) \land \text{Adjacent}([2,2],[2,3]) \Rightarrow \text{Stench}([2,3])$
  - ...
Existential instantiation

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

- Substitute new constant symbol \( k \) for variable \( v \) in \( \alpha \)
- E.g., \( \exists s,t \text{ At}(\text{Wumpus},s,t) \)
  - \( \text{At}(\text{Wumpus},S1,T1) \)
  - \( S1 \) and \( T1 \) are new constant symbols
- After removing quantifiers, perform PL inference
- Complete, but inefficient
- FOL is semi-decidable
  - Some algorithms can say yes to every entailed sentence, but no algorithm can say no to every non-entailed sentence
Substitution (binding) $\theta = \{x/y\}$

- Replace all occurrences of $x$ with $y$
- E.g., $\alpha = \text{At(Wumpus,s,t)}$, $\theta = \{s/[1,3], t/5\}$
  - $\alpha \theta = \text{At(Wumpus,[1,3],5)}$

Generalized Modus Ponens

$\frac{p'_1, p'_2, \ldots, p'_n, \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$

- where $\text{SUBST}(\theta,p'_i) = \text{SUBST}(\theta,p_i)$ for all $i$

Find $\theta$ via unification
Example

\[ \forall s, r \ Pit(s) \land \text{Adjacent}(s, r) \Rightarrow \text{Breeze}(r) \]

\[ \text{Pit}([3, 1]), \text{Adjacent}([3, 1], [2, 1]) \]

\[ p_1 = \text{Pit}(s), \ p_2 = \text{Adjacent}(s, r), \ q = \text{Breeze}(r) \]

\[ p_1' = \text{Pit}([3, 1]), \ p_2' = \text{Adjacent}([3, 1], [2, 1]) \]

\[ \theta = \{s/[3, 1], \ r/[2, 1]\} \]

\[ \text{SUBST}(\theta, q) = \text{Breeze}([2, 1]) \]
Unification

- Unification determines if two sentences match given some substitution (unifier)
- $\text{UNIFY}(p,q) = \emptyset$ where $\text{SUBST}(\emptyset, p) = \text{SUBST}(\emptyset, q)$
- Examples
  - $\text{UNIFY}(\text{At(Wumpus},s,t), \text{At(Wumpus,}[1,3],5)) = \{s/[1,3], t/5\}$
  - $\text{UNIFY}(\text{At(Wumpus},s,t), \text{At(Wumpus,}r,5)) = \{s/r, t/5\}$
  - $\text{UNIFY}(\text{At(Wumpus},s,t), \text{At(Wumpus,}\text{AgentLoc}(t),5)) = \{s/\text{AgentLoc}(t), t/5\} = \{s/\text{AgentLoc(5)}, t/5\}$
Standardizing apart
  - Use unique variable names in each sentence
  - \text{UNIFY} (\text{At}(x,[1,3],t), \text{At}(\text{Wumpus},x,t)) = \text{failure} \\
  - \text{UNIFY} (\text{At}(x_{17},[1,3],t), \text{At}(\text{Wumpus},x_{21},5)) = \\
    \{x_{17}/\text{Wumpus}, x_{21}/[1,3], t/5\}

Most General Unifier (MGU)
  - Unifier returned by Unify should place the least possible restrictions on variables \\
  - \text{UNIFY} (\text{Foo}(x,x),\text{Foo}([1,3],y)) = \{x/[1,3], y/[1,3]\} \text{ works} \\
  - But so does \{x/[1,3], y/x\} (more general)
Unification Algorithm

- Recursively compare two expressions
- Build up substitutions along the way

**Details**
- Compound expression of the form $F(A,B)$
- If $x = F(A,B)$, then $x.\text{OP} = F$, $x.\text{ARGS} = [A,B]$ (a list)
- List decomposed using First and Rest
- If $x = [A,B,C]$, then $x.\text{FIRST} = A$ and $x.\text{REST} = [B,C]$
function **UNIFY** \((x, y, \theta)\) returns a substitution to make \(x\) and \(y\) identical

**inputs:** \(x\), a variable, constant, list, or compound expression

\(y\), a variable, constant, list, or compound expression

\(\theta\), the substitution built up so far (optional, defaults to empty)

if \(\theta = \text{failure}\) then return failure

else if \(x = y\) then return \(\theta\)

else if VARIABLE?(\(x\)) then return UNIFY-VAR(\(x, y, \theta\))

else if VARIABLE?(\(y\)) then return UNIFY-VAR(\(y, x, \theta\))

else if COMPOUND?(\(x\)) and COMPOUND?(\(y\)) then

  return UNIFY(x.ARGS, y.ARGS, UNIFY(x.Op, y.Op, \(\theta\)))

else if LIST?(\(x\)) and LIST?(\(y\)) then

  return UNIFY(x.REST, y.REST, UNIFY(x.FIRST, y.FIRST, \(\theta\)))

else return failure
Unification

function `UNIFY-VAR (var, x, θ) returns a substitution
    if \{var / val\} ∈ θ then return `UNIFY(val, x, θ)
    else if \{x / val\} ∈ θ then return `UNIFY(var, val, θ)
    else if OCCUR-CHECK?(var; x) then return failure
    else return add \{var / x\} to θ

- Occur check
  - When matching variable and term, check if variable occurs in term
  - If so, failure; e.g., P(x) does not unify with P(P(x))
  - Makes `UNIFY quadratic in size of expression
  - Some inference systems omit occur check
Forward Chaining

- Start with atomic sentences in KB
- Apply Modus Ponens where possible to infer new atomic sentences
- Continue until goal is proven or no new inferences can be made
- Assume first-order definite clauses for now
  - Disjunction of literals with exactly one positive literal
  - E.g., $\forall x, y \left( \neg A(x) \lor \neg B(y) \lor C(x, y) \right)$
  - $\forall x, y A(x) \land B(y) \Rightarrow C(x, y)$
Example

- 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
"… it is a crime for an American to sell weapons to hostile nations."
- R1: $\forall x, y, z \ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)$

"… Nono, an enemy of America, …"
- R2: $\text{Enemy}(\text{Nono}, \text{America})$

"… Nono … has some missiles"
- R3: $\exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)$
- R4: $\text{Missile}(\text{M}_1)$
“… all of its missiles were sold to it by Colonel West”

- **R5**: \( \forall x \) Missile\( (x) \) \( \land \) Owns\( (Nono,x) \) \( \Rightarrow \) Sells\( (West,x,Nono) \)

“… Colonel West, who is American.”

- **R6**: American\( (West) \)

A few more rules…

- **R7**: \( \forall x \) Missile\( (x) \) \( \Rightarrow \) Weapon\( (x) \)
- **R8**: \( \forall x \) Enemy\( (x,America) \) \( \Rightarrow \) Hostile\( (x) \)
Example: Forward Chaining

R1: \{x_4/\text{West}, y_1/M_1, z_1/\text{Nono}\}

R2: \{x_3/\text{Nono}\}

R3: \{x_2/M_1\}

R4: \{x_1/M_1\}

R5: \{x_2/M_1\}

R6: \{x_1/M_1\}

American(West) R4 Missile(M_1) R3 Owns(Nono,M_1) R2 Enemy(Nono,America)

Weapon(M_1) R5 Sells(West,M_1,Nono) R8 Hostile(Nono)

Criminal(West)
function FOL-FC-Ask \((KB, \alpha)\) returns a substitution or false

inputs: \(KB\), the knowledge base, a set of first-order definite clauses
\(\alpha\), the query, an atomic sentence

local variables: \(new\), the new sentences inferred at each iteration

repeat until \(new\) is empty
\(new \leftarrow \{\}\)

for each rule in \(KB\) do
\((p_1 \land \ldots \land p_n \Rightarrow q) \leftarrow \text{STANDARDIZE-VARIABLES}(\text{rule})\)

for each \(\theta\) such that \(\text{SUBST}(\theta, p_1 \land \ldots \land p_n) = \text{SUBST}(\theta, p_1' \land \ldots \land p_n')\)
for some \(p_1', \ldots, p_n'\) in \(KB\)
\(q' \leftarrow \text{SUBST}(\theta, q)\)

if \(q'\) does not unify with some sentence already in \(KB\) or \(new\) then
add \(q'\) to \(new\)
\(\phi = \text{UNIFY}(q', \alpha)\)
if \(\phi\) is not fail then return \(\phi\)

add \(new\) to \(KB\)

return false
Forward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all known facts
    - Conjunct ordering
    - R5: \( \forall x \text{ Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \)
  - Recheck every rule on every iteration
    - Every new fact inferred on iteration \( t \) must be derived from at least one new fact inferred on iteration \( t-1 \)
    - Incremental forward chaining
  - Irrelevant facts (e.g., Enemy(Wumpus,America))
Backward Chaining

- Work backwards from the goal
- For rules concluding goal, add premises as new goals
- Continue until all open goals supported by known facts
- Again, assume first-order definite clauses for now
Example: Backward Chaining

1. **Criminal(West)**
   - R1: \(\{x_1/\text{West}\}\)

2. **American(West)**
   - R6

3. **Weapon(y)**
   - R7: \(\{x_2/y\}\)
     - \(\{y/M_1\}\)
     - **Missile(y)**
     - **Missile(M_1)**
     - R4

4. **Sells(West,M_1,z)**
   - R5: \(\{z/\text{Nono}\}\)
     - **Missile(M_1)**
     - **Owns(Nono, M_1)**
     - R3

5. **Hostile(Nono)**
   - R8

6. **Enemy(Nono,America)**
   - R2

Artificial Intelligence 80
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions

inputs: KB, a knowledge base

goals, a list of conjuncts forming a query (θ already applied)
θ, the current substitution, initially the empty substitution {}

local variables: answers, a set of substitutions, initially empty

if goals is empty then return {θ}
q' ← Subst(θ, First(goals))
for each sentence r in KB
where Standardize-Apart(r) = (p₁ \land \ldots \land p_n \Rightarrow q)
and θ' ← Unify(q, q') succeeds
    new_goals ← [p₁, \ldots, p_n | Rest(goals)]
    answers ← FOL-BC-Ask(KB, new_goals, Compose(θ', θ)) \cup answers
return answers
Backward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all open goals
    - More constraints
    - $R5: \forall x \text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})$
  - Recheck every rule on every iteration
    - Yes, but only those whose consequent unifies with an open goal
  - Irrelevant facts (e.g., $\text{Enemy}(\text{Wumpus},\text{America})$)
    - Excluded
Logic Programming

- A program is a set of logic statements defining the constraints of the problem
- Theorem-proving used to “run” the program
- Prolog is a logic programming language
  - Closed-world assumption
  - Supports arithmetic “X is 1+2” binds X to 3
  - Allows assertion and retraction of sentences
  - No occur check
  - Depth-first backward chaining (incomplete)
Prolog

- Uses uppercase for variables
- Uses lowercase for constants, functions and predicates
- Uses comma for AND
- Uses “:-” for implication (reversed)
- E.g., “A ∧ B ⇒ C” written as “C :- A, B.”
- sells(west,X,nono) :- missile(X), owns(nono,X).
- Gnu Prolog (www.gprolog.org)
Resolution using refutation (proof by contradiction) is sound and complete
- \( KB = \{ \neg A(x) \lor B(x), A(Wumpus) \} \)
- Prove: \( B(wumpus) \)
- Add negated goal to \( KB \): \( \neg B(wumpus) \)
- Search for contradiction using resolution
  - If result ever empty clause, then proven
- Resolve original clauses: \( B(wumpus) \)
- Resolve \( B(wumpus) \) and \( \neg B(wumpus) \): \( {} \)

Convert FOL to clausal form (CNF)

Efficient? Resolution strategies
Convert FOL to CNF

- Conjunctive Normal Form (CNF)
  - Conjunction of clauses
  - Each clause is a disjunction of literals
  - Variables assumed to be universally quantified

- Example
  - $\forall x,y,z \text{ American}(x) \land \text{ Weapon}(y) \land \text{ Sells}(x,y,z) \land \text{ Hostile}(z) \Rightarrow \text{ Criminal}(x)$
  - $\neg\text{American}(x) \lor \neg\text{Weapon}(y) \lor \neg\text{Sells}(x,y,z) \lor \neg\text{Hostile}(x) \lor \text{Criminal}(x)$

- Same as propositional logic, but need to eliminate existential quantifiers
Convert FOL to CNF

- **Step 1: Eliminate implications** ⇒
  - From: $\forall x \ A(x) \land B(x) \Rightarrow C(x)$
  - To: $\forall x \neg A(x) \lor \neg B(x) \lor C(x)$

- **Step 2: Move $\neg$ inwards**
  - $\neg \forall x \ A(x)$ becomes $\exists x \neg A(x)$
  - $\neg \exists x \ A(x)$ becomes $\forall x \neg A(x)$

- **Step 3: Standardize variables**
  - From: $(\forall x \ A(x)) \land (\forall x \ B(x))$
  - To: $(\forall x_1 \ A(x_1)) \land (\forall x_2 \ B(x_2))$
Step 4: Skolemize (Skolemization)

- Eliminate existential quantifiers by replacing them with a new constant or function
- Skolem constant, Skolem function
- Arguments of the Skolem function are all the universally quantified variables in whose scope the existential quantifier appears

- From: $\exists x \ P(x)$, To: $P(F1)$
- From: $\forall x,y \ \exists z \ P(x,y,z)$
- To: $\forall x,y \ P(x,y,F1(x,y))$

Thoralf Skolem (1887–1963)
Norwegian mathematician
Convert FOL to CNF

- **Step 5:** Drop universal quantifiers
  - All remaining variables universally quantified
  - So, just drop the ∀x,y,…

- **Step 6:** Distribute ∨ over ∧
  - From: \((A(x) \land B(x)) \lor C(x)\)
  - To: \((A(x) \lor C(x)) \land (B(x) \lor C(x))\)
Example (FOL $\rightarrow$ CNF)

- What is a brick?
  - A brick is on something that is not a pyramid
  - There is nothing that a brick is on and that is on the brick as well
  - There is nothing that is not a brick and also is the same thing as a brick.

$$\forall x \ [\text{Brick}(x) \ \Rightarrow \ (\exists y \ [\text{On}(x,y) \ \land \ \neg \text{Pyramid}(y)] \ \land$$

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

$$\forall y \ [\neg \text{Brick}(y) \ \Rightarrow \ \neg \text{Equal}(x,y)]]$$
Example (FOL $\rightarrow$ CNF)

- Step 1: Eliminate implications

\[ \forall x \ [\neg \text{Brick}(x) \lor (\exists y \ [\text{On}(x,y) \land \neg \text{Pyramid}(y)] \land \\
\neg \exists y \ [\text{On}(x,y) \land \text{On}(y,x)] \land \\
\forall y \ [\neg \neg \text{Brick}(y) \lor \neg \text{Equal}(x,y))]) \]
Example (FOL $\rightarrow$ CNF)

- Step 2: Move $\neg$ inwards

\[
\forall x \ (\neg\text{Brick}(x) \lor (\exists y \ [\text{On}(x,y) \land \neg\text{Pyramid}(y)] \land \\
\forall y \ [\neg\text{On}(x,y) \lor \neg\text{On}(y,x)] \land \\
\forall y \ [\text{Brick}(y) \lor \neg\text{Equal}(x,y)])
\]

\[
\forall x \ (\neg\text{Brick}(x) \lor (\exists y \ [\text{On}(x,y) \land \neg\text{Pyramid}(y)] \land \\
\forall y \ [\neg\text{On}(x,y) \lor \neg\text{On}(y,x)] \land \\
\forall y \ [\text{Brick}(y) \lor \neg\text{Equal}(x,y)])
\]
Step 3: Standardize variables

\[
\forall x \ [\neg \text{Brick}(x) \lor (\exists y \ [\text{On}(x,y) \land \neg \text{Pyramid}(y)]) \land \\
\forall a \ [\neg \text{On}(x,a) \lor \neg \text{On}(a,x)] \land \\
\forall b \ [\text{Brick}(b) \lor \neg \text{Equal}(x,b)]
\]
Example (FOL $\rightarrow$ CNF)

- Step 4: Skolemization

$$\forall x [\neg \text{Brick}(x) \lor ([\text{On}(x,F(x)) \land \neg \text{Pyramid}(F(x))] \land \forall a [\neg \text{On}(x,a) \lor \neg \text{On}(a,x)] \land \forall b [\text{Brick}(b) \lor \neg \text{Equal}(x,b)]]$$

- Step 5: Drop universal quantifiers

$$\neg \text{Brick}(x) \lor ([\text{On}(x,F(x)) \land \neg \text{Pyramid}(F(x))] \land [\neg \text{On}(x,a) \lor \neg \text{On}(a,x)] \land [\text{Brick}(b) \lor \neg \text{Equal}(x,b)]]$$
Example (FOL $\to$ CNF)

- Step 6: Distribute $\lor$ over $\land$

$$(\neg\text{Brick}(x) \lor \text{On}(x,F(x))) \land$$

$$(\neg\text{Brick}(x) \lor \neg\text{Pyramid}(F(x))) \land$$

$$(\neg\text{Brick}(x) \lor \neg\text{On}(x,a) \lor \neg\text{On}(a,x)) \land$$

$$(\neg\text{Brick}(x) \lor \text{Brick}(b) \lor \neg\text{Equal}(x,b))$$
Resolution Inference Rule

\[ l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_n \]

\[
\text{SUBST}(\theta, l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)
\]

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

- Resolution plus proof by refutation is sound and complete
Example Proof: Criminal(West)

CNF
- \neg \text{American}(x) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(x,y,z) \lor \neg \text{Hostile}(z) \lor \text{Criminal}(x)
- \neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono})
- \neg \text{Missile}(x) \lor \text{Weapon}(x)
- \neg \text{Enemy}(x,\text{America}) \lor \text{Hostile}(x)
- \text{Enemy}(\text{Nono},\text{America})
- \text{Owns}(\text{Nono},M_1)
- \text{Missile}(M_1)
- \text{American}(\text{West})

Prove: Criminal(West)
- Add \neg \text{Criminal}(\text{West}) to KB and derive empty clause
Example Proof: Criminal(West)

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

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

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

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

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

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

\[ \text{Missile}(M_1) \]

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

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

\[ \neg \text{Sells}(\text{West},M_1,z) \lor \neg \text{Hostile}(z) \]

\[ \text{Missile}(M_1) \]

\[ \neg \text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \text{Owns}(\text{Nono},M_1) \]

\[ \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \]

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

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

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

\[ \neg \text{Enemy}(\text{Nono},\text{America}) \]
Prove: $\exists c \text{ Criminal}(c)$

- Add $\neg \exists c \text{ Criminal}(c)$ to KB
- I.e., add $\neg \text{Criminal}(c)$ to KB

Generated clauses

- $\neg \text{American}(c) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(c,y,z) \lor \neg \text{Hostile}(z)$
- $\neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)$
- $\neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z)$
- $\neg \text{Sells}(\text{West},M_1,z) \lor \neg \text{Hostile}(z)$
- $\neg \text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono})$
- $\neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono})$
- $\neg \text{Hostile}(\text{Nono})$
- $\neg \text{Enemy}(\text{Nono},\text{America})$
Add clause with negated goal and answer literal to KB

Search for clause containing only answer literal

Prove: $\exists x, y, z \text{ Goal}(x,y,z)$

Add $(\neg \text{Goal}(x,y,z) \lor \text{Answer}(x,y,z))$ to KB

Final clause $\text{Answer}(x,y,z)$ will have variables bound to answers
Who is the Criminal?

- Prove: $\exists c \text{ Criminal}(c)$ and retrieve $c$
  - Add $[\neg \text{Criminal}(c) \lor \text{Answer}(c)]$ to KB

- Generated clauses
  - $\neg \text{American}(c) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(c, y, z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(c)$
  - $\neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West}, y, z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West})$
  - $\neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West}, y, z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West})$
  - $\neg \text{Sells}(\text{West}, M_1, z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West})$
  - $\neg \text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono}, M_1) \lor \neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West})$
  - $\neg \text{Owns}(\text{Nono}, M_1) \lor \neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West})$
  - $\neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West})$
  - $\neg \text{Enemy}(\text{Nono}, \text{America}) \lor \text{Answer}(\text{West})$
  - $\text{Answer}(\text{West})$
Equality

- Handle \((x = y)\) terms in resolution theorem proving
- Let \(\text{SUB}(x,y,m)\) mean to replace \(x\) with \(y\) everywhere \(x\) occurs within \(m\)
- Demodulation

\[
\frac{x = y, \quad m_1 \lor \cdots \lor m_n}{\text{SUB} (\text{SUBST}(\theta, x), \text{SUBST}(\theta, y), m_1 \lor \cdots \lor m_n )}
\]

where \(\text{UNIFY}(x, z) = \theta\) and \(z\) appears in literal \(m_i\)

- Example
  - Given: \(\text{Father}(\text{Father}(x)) = \text{Grandfather}(x)\)
    \(\text{Birthdate}(\text{Father}(\text{Father}(\text{Bob})), 1941)\)
  - Infer: \(\text{Birthdate}(\text{Grandfather}(\text{Bob}), 1941)\)
Equality

- Paramodulation

\[ l_1 \lor \ldots \lor l_k \lor x = y, \quad m_1 \lor \cdots \lor m_n \]

\[
\text{SUB(SUBST}(\theta, x), \text{SUBST}(\theta, y), \text{SUBST}(\theta, l_1 \lor \ldots \lor l_k \lor m_1 \lor \cdots \lor m_n))
\]

where \( \text{UNIFY}(x, z) = \theta \) and \( z \) appears in literal \( m_i \)

- Paramodulation is complete for FOL with equality
Strategies for Efficient Resolution

- **Unit preference**
  - Prefer resolutions where one clause contains a single literal (unit clause)
  - New sentence always shorter
  - Incomplete (but complete for Horn clauses)

- **Set of support**
  - Subset of clauses, one of which must always be used for resolution
  - New clauses added to set of support
  - Complete if remaining clauses satisfiable
  - Initially, SoS = negated goal
Strategies for Efficient Resolution

- **Input resolution**
  - Every resolution combines sentence from (KB+goal) and some other sentence
  - E.g., Criminal(West) proof
  - Incomplete (but complete for Horn clauses)

- **Linear resolution**
  - Input resolution, but also allowing ancestor sentences
  - Complete

- **Subsumption**
  - Eliminate sentences more specific than (subsumed by) others
  - E.g., if P(x) in KB, then don’t add P(A)
Theorem Proving: State of the Art

- Vampire (vprover.github.io)
- iProver (www.cs.man.ac.uk/~korovink/iprover)
- Conference on Automated Deduction (CADE) ATP System Competition (CASC)
  - www.cs.miami.edu/~tptp/CASC/
- Applications
  - Mathematical theorem proving
  - Hardware and software verification and synthesis
Summary

- Knowledge-based agent
- Logic
- Propositional logic
- First-order logic
- Inference
  - Unification
  - Resolution
  - Proof by contradiction