Logic

Larry Holder
School of EECS
Washington State University
Knowledge-based Agent

- Knowledge base
  - Tell agent about the environment

- Knowledge representation
  - First-order logic
  - Many others…

- Reasoning via inference
  - Ask agent how to achieve goal based on current knowledge
function **KB-A GENT** (*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*
Acting Logically in Wumpus World

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
Acting Logically in Wumpus World

4
3
2
1

4 1 2 3 4
3 1 2 3 4
2 1 2 3 4
1 1 2 3 4
Acting Logically in Wumpus World

- Percept\(_1\) = [None, None, None, None, None, None]
  - [1,2] and [2,1] safe
- Action = GoForward
- Percept\(_2\) = [None, Breeze, None, 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

**Example**
- Syntax: Wumpus(2,2)
- Semantics: Wumpus is in location (2,2)
Logical inference is the process of inferring one sentence is true from others.

Inference should be **sound** or **truth-preserving**
- Everything inferred is true.

Inference should be **complete**
- Everything that is true can be inferred.
Different Logics

- **Propositional logic** assumes world consists of facts that are either true, false or unknown
  - E.g., \( Wumpus(1,3) \Rightarrow Stench(1,2) \)

- **First-order logic** assumes world consists of facts, objects and relations that are either true, false or unknown
  - E.g., \( Wumpus(x,y) \land Adjacent(x,y,w,z) \Rightarrow Stench(w,z) \)

- **Temporal logic** = FOL where facts hold at particular times
  - E.g., \( \text{Before(Action(Shoot), Percept(Scream))} \)

- **Higher-order logic** assumes world includes first-order relations as objects
  - E.g., \( \text{Know( [ Wumpus(x,y) \land Adjacent(x,y,w,z) \Rightarrow Stench(w,z) ] )} \)

- **Probabilistic logic** = propositional logic with a degree of belief for each fact
  - E.g., \( P(Wumpus(1,3)) = 0.067 \)
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))
FOL Syntax

Sentence → AtomicSentence | ComplexSentence

AtomicSentence → Predicate | Predicate (Term,...) | Term = Term

ComplexSentence → (Sentence) | [Sentence]
| ¬ Sentence
| Sentence ∧ Sentence
| Sentence ∨ Sentence
| Sentence ⇒ Sentence
| Sentence ⇔ Sentence
| Quantifier Variable,... Sentence

Term → Function (Term,...) | Constant | Variable

Quantifier → ∀ | ∃

Constant → A | B | Wumpus | 1 | 2 | ...

Variable → a | x | s | ...

Predicate → True | False | Adjacent | At | Alive | ...

Function → Location | RightOf | ...

Operator Precedence: ¬, =, ∧, ∨, ⇒, ⇔
FOL Syntax

- Not (\(\neg\)) is a **negation**
- Literal is either an atomic sentence (**positive literal**) or a negated atomic sentence (**negative literal**)
  - \(\neg\text{Breeze}(1,1), \text{Breeze}(2,1)\)
- And (\(\land\)) is a **conjunction**; its parts are **conjuncts**
- Or (\(\lor\)) is a **disjunction**; its parts are **disjuncts**
- Implies (\(\Rightarrow\)) is an **implication**
  - \(\text{Pit}(2,2) \Rightarrow \text{Breeze}(1,2)\)
  - Lefthand side is the **antecedent** or **premise**
  - Righthand side is the **consequent** or **conclusion**
- If and only if (\(\Leftrightarrow\)) is a **biconditional**
How to determine the truth value (true or false) of every sentence

True is always true
False is always false

Truth values of every other sentence must be specified directly or inferred

- E.g., \texttt{Wumpus(2,2)} is true, \texttt{Wumpus(3,3)} is false
Semantics for complex sentences
- \( \neg P \) is true iff \( P \) is false
- \( P \land Q \) is true iff both \( P \) and \( Q \) are true
- \( P \lor Q \) is true iff either \( P \) or \( Q \) is true
- \( P \Rightarrow Q \) is true unless \( P \) is true and \( Q \) is false
- \( P \iff Q \) is true iff \( P \) and \( Q \) are both true or both false

### Truth Table

<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 \iff 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>

Artificial Intelligence 17
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 \(\forall x,y \neg At(Wumpus,x,y) \lor Stench(x+1,y)\)
- \(\forall x \ P(x) \equiv P(A) \land P(B) \land P(Wumpus) \land \ldots\)
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) ∧ Wumpus(w)
  - Same as ∃ w,x,y At(w,x,y) ⇒ Wumpus(w) ?

- ∃x P(x) ≡ P(A) ∨ P(B) ∨ P(Wumpus) ∨ ...
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 $\forall y \ \exists x$ same as $\exists x, y$

- $\exists x \ \forall y \ \text{Likes}(x, y)$
- $\forall y \ \exists x \ \text{Likes}(x, y)$
- $\forall x \ \exists y \ \text{Likes}(x, y)$
- $\exists y \ \forall x \ \text{Likes}(x, y)$
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** \((\text{Term}_1 = \text{Term}_1)\) means \(\text{Term}_1\) and \(\text{Term}_2\) refer to the same object
  - E.g., \(\text{RightOf}(\text{Location}(1,1)) = \text{Location}(2,1)\)
- 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

- 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

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

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

- **ASKVARS (KB, α)**
  - Returns answers (variable bindings) that make α true
  - Or, use Answer literal (later)
  - ASK (KB, ∃a Action(a, 5) ^ 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”
  - $\forall t, s, g, m, c \text{ Percept}([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$
  - $\forall t, s, b, m, c \text{ Percept}([s, b, Glitter, m, c], t) \Rightarrow Glitter(t)$

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

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

- 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
    \[ \forall t \ \text{At}(\text{Wumpus}, [1,3], t) \]
- Nothing can be in two places at once
  \[ \forall x,s_1,s_2,t \ \text{At}(x, s_1, t) \land \text{At}(x, s_2, t) \Rightarrow s_1 = s_2 \]
- Successor-state axioms for each action
  - Describes what’s true before and after action
    \[ \forall t \ \text{HaveArrow}(t+1) \iff (\text{HaveArrow}(t) \land \neg \text{Action(Shoot, t)}) \]
    \[ \forall t \ \text{HaveGold}(t+1) \iff (\text{HaveGold}(t) \lor (\text{Glitter}(t) \land \text{Action(Grab, t)})) \]
    \[ ... \]
FOL for the Wumpus World

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

- Only one arrow?

- Only one gold?
- At least one pit?
function HYBRID-WUMPUS-AGENT(\textit{percept}) \textbf{returns} an action

\textbf{inputs:} \textit{percept}, a list, [\textit{stench, breeze, glitter, bump, scream}]

\textbf{persistent:} \(KB\), a knowledge base, initially the atemporal “wumpus physics”
\(t\), a counter, initially 0, indicating time
\(plan\), an action sequence, initially empty

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

\textbf{if} \ \text{ASK}(\textit{KB}, \text{Glitter}^{t}) = \text{true} \ \textbf{then}
\(plan \leftarrow [\text{Grab}] + \text{PLAN-ROUTE(\textit{current}, \{[1,1]\}, \text{safe})} + [\text{Climb}]\)

\textbf{if} \ \text{plan} \ \text{is empty} \ \textbf{then}
\(unvisited \leftarrow \{[x, y] : \text{ASK}(KB, \text{L}^{t'}_{x,y}) = \text{false} \text{ for all } t' \leq t\}\)
\(plan \leftarrow \text{PLAN-ROUTE(\textit{current}, \text{unvisited} \cap \text{safe}, \text{safe})}\)

\textbf{if} \ \text{plan} \ \text{is empty and} \ \text{ASK}(\textit{KB}, \text{HaveArrow}^{t}) = \text{true} \ \textbf{then}
\(\text{possible\_wumpus} \leftarrow \{[x, y] : \text{ASK}(KB, \neg \text{W}_{x,y}) = \text{false}\}\)
\(plan \leftarrow \text{PLAN-SHOT(\textit{current}, \text{possible\_wumpus}, \text{safe})}\)

\textbf{if} \ \text{plan} \ \text{is empty} \ \text{then} \quad / / \text{no choice but to take a risk}
\(\text{not\_unsafe} \leftarrow \{[x, y] : \text{ASK}(KB, \neg \text{OK}^{t}_{x,y}) = \text{false}\}\)
\(plan \leftarrow \text{PLAN-ROUTE(\textit{current}, \text{not\_unsafe} \cap \text{safe}, \text{safe})}\)

\textbf{if} \ \text{plan} \ \text{is empty} \ \textbf{then}
\(plan \leftarrow \text{PLAN-ROUTE(\textit{current}, \{[1, 1]\}, \text{safe})} + [\text{Climb}]\)
\(action \leftarrow \text{POP(\textit{plan})}\)
\text{TELL}(\textit{KB}, \text{MAKE-ACTION-SENTENCE(\textit{action}, \textit{t})})
\(t \leftarrow t + 1\)
\textbf{return} \ \textit{action}
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 SEARCH(problem)    // Any search algorithm
Now that we have FOL, how can we perform sound, complete and efficient inference?

Approaches
- Generalized modus ponens
- Forward and backward chaining
- Resolution

State of the art
Inference in First-Order Logic

- Carefully…

Monty Python and the Holy Grail (1975)
Inference Rules

- Notation

\[
\text{sentences given} \quad \frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}
\]

- Modus Ponens
Generalized Modus Ponens

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

- Generalized Modus Ponens
  \[
  \frac{p_1', p_2', \ldots, p_n'}{(p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}
  \][SUBST($\theta$, $q$)]

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

- Find $\theta$ via unification
Example

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

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

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

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

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

\[ \text{SUBST}(\theta, q) = \text{Breeze}([2,1]) \]
Unification determines if two sentences match given some substitution (unifier)

\[ \text{UNIFY}(p, q) = \emptyset \text{ where } \text{SUBST}(\emptyset, p) = \text{SUBST}(\emptyset, q) \]

Examples

- \[ \text{UNIFY} (\text{At}(\text{Wumpus}, s, t), \text{At}(\text{Wumpus}, [1,3], 5)) = \{s/[1,3], t/5\} \]
- \[ \text{UNIFY} (\text{At}(\text{Wumpus}, s, t), \text{At}(\text{Wumpus}, r, 5)) = \{s/r, t/5\} \]
- \[ \text{UNIFY} (\text{At}(\text{Wumpus}, s, t), \text{At}(\text{Wumpus}, \text{AgentLoc}(t), 5)) = \{s/\text{AgentLoc}(t), t/5\} = \{s/\text{AgentLoc}(5), t/5\} \]
Unification

- Standarize variables apart
  - Use unique variable names in each sentence
    - $\text{UNIFY} \ (\text{At}(x,[1,3],t), \text{At}(\text{Wumpus},x,t)) = \text{failure}$
    - $\text{UNIFY} \ (\text{At}(x_{17},[1,3],t), \text{At}(\text{Wumpus},x_{21},5)) = \{x_{17}/\text{Wumpus}, \ x_{21}/[1,3], \ t/5\}$
Unification

- 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 \textsc{Unify} quadratic in size of expression
  - Some inference systems omit occur check
## Unification Examples

<table>
<thead>
<tr>
<th>Term 1</th>
<th>Term 2</th>
<th>Substitution (or fail)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glitter($x,y$)</td>
<td>Glitter(3,3)</td>
<td></td>
</tr>
<tr>
<td>Adjacent($x,2,2,y$)</td>
<td>Adjacent(2,$y$,x,3)</td>
<td></td>
</tr>
<tr>
<td>At($w,x,y$)</td>
<td>At(Wumpus,$u,3$)</td>
<td></td>
</tr>
<tr>
<td>At(Agent,$x$,$\text{Row}(\text{Wumpus})$)</td>
<td>At($z,3,\text{Row}(w)$)</td>
<td></td>
</tr>
<tr>
<td>At($w$,\text{Column}($w$),$y$)</td>
<td>At(Agent,\text{Column}(\text{Wumpus}),3)</td>
<td></td>
</tr>
</tbody>
</table>
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., $\forall x,y \neg A(x) \lor \neg B(y) \lor C(x,y) \equiv \forall x,y A(x) \land B(y) \Rightarrow C(x,y)$
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”
\[ R3: \exists x \text{ Owns}(\text{Nono},x) \land \text{Missile}(x) \]
\[ R4: \text{Missle}(\text{M}_1) \]

Existential Instantiation
“… all of its missiles were sold to it by Colonel West”
- **R5**: \( \forall x \text{ Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono}) \)

“… Colonel West, who is American.”
- **R6**: \( \text{American}(\text{West}) \)

A few more rules…
- **R7**: \( \forall x \text{ Missile}(x) \Rightarrow \text{Weapon}(x) \)
- **R8**: \( \forall x \text{ Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x) \)
Example: Forward Chaining

\[ R_6: \{x_4/$\text{West}$, y_1/$M_1$, z_1/$\text{Nono}$\} \]

American($\text{West}$) → Missile($M_1$) → Owns($\text{Nono}, M_1$) → Enemy($\text{Nono}, \text{America}$)

Weapon($M_1$) → Sells($\text{West}, M_1, \text{Nono}$) → Hostile($\text{Nono}$)

Criminal($\text{West}$)
Forward Chaining

- Sound?
- Complete?
- Efficient?
  - Matching all rules against all known facts
    - Conjunct ordering
    - $R5: \forall x \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})$
  - 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., $\text{Enemy}(\text{Wumpus}, \text{America})$)
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

- **Criminal(West)**
  - **American(West)**
    - **Weapon(y)**
      - **Sells(West,M_1,z)**
        - **Hostile(Nono)**
          - **Enemy(Nono,America)**
  - **Missile(y)**
    - **Missile(M_1)**
      - **Owns(Nono, M_1)**
  - **Missile(M_1)**

Rules:
- **R1**: \{x_1/West\}
- **R2**: \{y/M_1\}
- **R3**: \{z/Nono\}
- **R4**: \{y/M_1\}
- **R5**: \{z/Nono\}
- **R6**: \{x_1/West\}
- **R7**: \{x_2/y\}
- **R8**:
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 in Prolog
Resolution Inference Rule

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

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

- \( l's \) and \( m's \) are literals
- A clause is a disjunction of literals
- Resolution takes two clauses and infers a new clause
Resolution using refutation (proof by contradiction) is sound and complete

- $\text{KB} = \{\neg A(x) \lor B(x), A(\text{Wumpus})\}$
- Prove: $B(\text{wumpus})$
- Add negated goal to $\text{KB}$: $\neg B(\text{wumpus})$
- Search for contradiction using resolution
  - If result ever empty clause, then proven
- Resolve original clauses: $B(\text{wumpus}), \theta = \{x/\text{wumpus}\}$
- Resolve $B(\text{wumpus})$ and $\neg B(\text{wumpus})$: $\{\}$

Convert FOL to clausal form (CNF)

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

Example
- ∀x, y, z American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
- ¬American(x) ∨ ¬Weapon(y) ∨ ¬Sells(x, y, z) ∨ ¬Hostile(x) ∨ Criminal(x)
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: \((A(x) \lor C(x)) \land (B(x) \lor C(x))\)
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.

∀x [Brick(x) ⇒ (∃y [On(x,y) ∧ ¬Pyramid(y)] ∧ ¬∃y [On(x,y) ∧ On(y,x)] ∧ ∀y [¬Brick(y) ⇒ ¬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

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

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

- Step 3: Standardize variables

$$\forall x [\neg Brick(x) \lor (\exists y [On(x,y) \land \neg Pyramid(y)]) \land$$
$$\forall a [\neg On(x,a) \lor \neg On(a,x)] \land$$
$$\forall b [Brick(b) \lor \neg Equal(x,b)]]$$
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**

$$
\neg \text{Brick}(x) \lor ([\text{On}(x,F(x)) \land \neg \text{Pyramid}(F(x))] \land \\
[\neg \text{On}(x,a) \lor \neg \text{On}(a,x)] \land \\
[\text{Brick}(b) \lor \neg \text{Equal}(x,b)])
$$
Step 6: Distribute $\lor$ over $\land$

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

\[
-\text{American}(x) \lor -\text{Weapon}(y) \lor -\text{Sells}(x,y,z) \lor -\text{Hostile}(z) \lor \text{Criminal}(x) \\
-\text{Criminal}(\text{West}) \\
\text{American}(\text{West}) \lor -\text{Weapon}(y) \lor -\text{Sells}(\text{West},y,z) \lor -\text{Hostile}(z) \\
-\text{Missile}(x) \lor \text{Weapon}(x) \\
\text{Missile}(M_1) \lor -\text{Missile}(y) \lor -\text{Sells}(\text{West},y,z) \lor -\text{Hostile}(z) \\
-\text{Missile}(x) \lor -\text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono}) \\
-\text{Sells}(\text{West},M_1,z) \lor -\text{Hostile}(z) \\
\text{Missile}(M_1) \lor -\text{Missile}(M_1) \lor -\text{Owns}(\text{Nono},M_1) \lor -\text{Hostile}(\text{Nono}) \\
-\text{Owns}(\text{Nono},M_1) \lor -\text{Hostile}(\text{Nono}) \\
-\text{Enemy}(x,\text{America}) \lor \text{Hostile}(x) \\
-\text{Hostile}(\text{Nono}) \\
\text{Enemy}(\text{Nono},\text{America}) \lor -\text{Enemy}(\text{Nono},\text{America})
\]
Is There a Criminal?

- **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: \( \exists c \) Criminal(c) and retrieve c
  - Add \([ \neg \text{Criminal}(c) \lor \text{Answer}(c) ]\) to KB
- Generated clauses
  - \( \neg \text{American}(c) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(c,y,z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(c) \)
  - \( \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Missile}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Sells}(\text{West},M_1,z) \lor \neg \text{Hostile}(z) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Missile}(M_1) \lor \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Owns}(\text{Nono},M_1) \lor \neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Hostile}(\text{Nono}) \lor \text{Answer}(\text{West}) \)
  - \( \neg \text{Enemy}(\text{Nono},\text{America}) \lor \text{Answer}(\text{West}) \)
  - \( \text{Answer}(\text{West}) \)
Another Example

- Willy is a wumpus. Swiper is not a wumpus, and Swiper is not a dog. If something is neither a wumpus nor a dog, then it is a fox. Foxes jump over Wumpuses.
- Prove that Swiper jumps over Willy.
Another Example (cont.)
Another Example (cont.)
Theorem Proving: State of the Art

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

- Knowledge-based (logical) agent
- First-order logic
- Inference
  - Resolution proof by refutation